

TRADE-OFFS BETWEEN DEPTH AND WIDTH IN  
PARALLEL COMPUTATION\*

UZI VISHKIN† AND AVI WIGDERSON‡

**Abstract.** A new technique for proving lower bounds for parallel computation is introduced. This technique enables us to obtain, for the first time, nontrivial tight lower bounds for shared-memory models of parallel computation that allow several processors to have simultaneous access to the same memory location. Specifically, we use a concurrent-read concurrent-write model of parallel computation. It has  $p$  processors, each has access to a common memory of size  $m$  (also called *communication width* or *width* in short). The input to the problem is located in an additional read-only portion of the common memory.

For a wide variety of problems (including parity, majority and summation) we show that the time complexity  $T$  (depth) and the communication width  $m$  are related by the trade-off curve  $mT^2 = \Omega(n)$ , (where  $n$  is the size of the input), *regardless* of the number of processors. Moreover, for every point on this curve with  $m = O(n/\log^2 n)$  we give a matching upper bound with the *optimal* number of processors.

We extend our technique to prove  $mT^3 = \Omega(n)$  trade-off for a class of "simpler" functions (including Boolean OR) on a weaker model that forbids simultaneous write access. We also state and give a proof of a new result by Beame [B-83] that achieves a tight lower bound for the OR in this model, namely  $mT^2 = \Omega(n)$ . These results improve the lower bound of Cook and Dwork [CD-82] when communication is limited.

**Key words.** synchronous parallelism, parallel time complexity, communication width, trade-offs between complexity measures, lower bounds

**1. Introduction.** Consider the following informal problem: there are a large number of people (or processing units), each knows  $n$  numbers  $a_1, a_2, \dots, a_n$ . They all wish to compute the sum of these numbers. If they cannot communicate, there is no way to avoid sequential ( $\Omega(n)$  time) summation by each person separately. On the other hand, it is shown in the paper that with only one communication channel (one cell of shared memory) this time can be reduced to  $O(\sqrt{n})$ . With  $n$  (resp.  $2^n$ ) shared memory cells the time can be reduced further to  $O(\log n)$  (resp.  $O(1)$ ). This exemplifies that a communication facility is essential for any utilization of parallelism, and that its size directly affects the performance of the algorithm.

The size of the common memory required by a given parallel algorithm will be determined by two principal factors.

(a) *Input availability.* The size of the input, in the case that the input is placed in the common memory, or the need to transfer input data in the case that the input is initially distributed among the local memories.

(b) *Cooperation between processors.* The transmission of intermediate results between processors, utilized to obtain fast processing time.

Here we propose to concentrate on point (b). For this reason we put the input in a "read only" common memory.

In this paper we will concentrate on parallel RAMs (PRAMs), in particular, the Concurrent-Read Concurrent-Write PRAM (CRCW PRAM) and the Concurrent-

\* Received by the editors May 17, 1983, and in revised form November 15, 1983. This paper is based on *Trade-offs between depth and width in parallel computation* by U. Vishkin and A. Wigderson, appearing in the 24th Annual Symposium on Foundations of Computer Science, November 7-9, 1983, Tucson, AZ, pp. 146-153. © 1983 IEEE.

† Department of Computer Science, School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel. This research was conducted while this author was at Courant Institute, New York University, New York, New York.

‡ Computer Science Division, University of California, Berkeley, California 94720. This research was conducted while this author was in the Electrical Engineering and Computer Science Departments, Princeton University, Princeton, New Jersey.

Read Exclusive-Write PRAM (CREW PRAM). Both models are precisely defined in § 2. In the above models processors communicate via a shared memory. Therefore the size of the communication facility of the machine, here called *communication width* (or *width* in short) is simply the number of shared memory cells. We consider the width  $m$  a resource, together with the size  $p$  (the number of processors) and the depth  $T$  (the running time), and we seek trade-offs between the three.

One of the subtleties in proving lower bounds for these models, is that information may be communicated by the fact that no processor writes into a common memory cell. We introduce a novel technique to deal with this difficulty.

For a large class of functions, which includes Parity and Majority, we prove  $T = \Omega((n/m)^{1/2})$  on the CRCW PRAM, where  $n$  is the size of the input. This lower bound is tight for all values of width  $m = O(n/\log^2 n)$ . This is the first time nontrivial tight lower bounds are achieved for a model that allows concurrent write access. The only known lower bound on the CRCW PRAM model is given in Stockmeyer and Vishkin [SV-82]. They show, using a result of Furst, Saxe, and Sipser [FSS-81], that it is impossible to compute parity in this model in constant time using a polynomial number of processors. There is, however, a large gap between this lower bound and the best upper bound known for a polynomial number of processors, which is  $O(\log n/\log \log n)$ . (See [CSV-82]).

For another class of functions, which includes the functions AND and OR, we prove a lower bound of  $T = \Omega((n/m)^{1/3})$  on the CREW PRAM. This lower bound extends the  $\Omega(\log n)$  of Cook and Dwork for small values of  $m$ , and further discerns the power of CRCW PRAM from the CREW PRAM. At this point we state, and give a proof, of a new result by Beame that achieves a tight lower bound for computing the OR in this model. For a different class of functions (that include OR) he proves  $T = \Omega((n/m)^{1/2})$ .

Both our lower bounds hold regardless of the number of processors, while the upper bounds are achieved with the smallest possible number of processors.

Our study of values of  $m$  which are smaller than input size requires us to add a read only input tape to the model, as is done in the study of space bounded Turing machines. The interest in those values is not solely theoretical—it is well founded in practice. For example the "Ethernet" can be considered as a PRAM with only one shared memory cell. Also, the papers Gottlieb et al. [GGKMRS-82], Kuck [K-77] and Vishkin [V-82] imply that minimizing the size of shared memory (that can be accessed in parallel) may amount to hardware feasibility of the parallel machine.

The paper is organized as follows: precise definitions and the lower bounds are given in § 2. Section 3 contains the upper bounds and § 4 concludes the paper and suggests further research directions. To improve the readability of § 2, some of the proofs were deferred to the appendix.

**2. Lower bounds.** In the first subsection we give precise definitions of the models of computation when the communication width  $m = 1$ , and of the types of functions we are interested in. Subsections 2.2 and 2.3 contain the lower bound proofs for the concurrent-write and exclusive-write models respectively when  $m = 1$ . In the last subsection we show how to extend the lower bounds for arbitrary communication width.

**2.1. Definitions.**

**DEFINITION 2.1.** A CRCW PRAM (1) consists of a set  $\Pi = \{p_1, p_2, \dots\}$  of processors, a number  $n$  of inputs,  $n$  read-only input cells  $X(1), X(2), \dots, X(n)$ , one common memory cell  $C$ , an alphabet  $\Sigma$  and an execution time  $T$ .

Each processor  $p_i$  has  
 $p_i: Q_i \rightarrow \{1, 2, \dots, n\}$   
 $\sigma_i: Q_i \rightarrow \Sigma$ -the symbol  
 $\delta_i: Q_i \times \Sigma \times \Sigma \rightarrow Q_i$ -the

At each time period  $t$  cell  $C$  contains a symbol  $s^t \in \Sigma$  ( $1 \leq i \leq n$ ), the cell  $p_i$  is in an initial state  $q_i^0 \in Q_i$

$$q_i^{t+1} = \delta_i(q_i^t, \sigma_i(s^t), \sigma_i(s^t))$$

$$s^{t+1} = \begin{cases} s^t & \text{if } \sigma_i(s^t) = \sigma_i(s^t) \\ \sigma_i(s^t) & \text{if } \sigma_i(s^t) \neq \sigma_i(s^t) \end{cases}$$

The value  $f(x_1, x_2, \dots)$  is said to be computed if the contents  $s^T$  of  $C$  at time  $T$  is  $f(x_1, x_2, \dots)$ .

**DEFINITION 2.2.** A CRCW PRAM (1), with only one exception that writes, i.e. at most one processor writes in each time period.

**Remarks.** These lower bounds are achieved for the processors be nonuniformly. Also note that we use the contents of  $C$ , (and in the case of doing so).

**DEFINITION 2.3.** Let  $\Sigma$  be an alphabet. A function  $f: \mathbf{I}^k \rightarrow \Sigma$  is said to be  $k$ -sensitive w.r.t.  $f$ , if for all  $j \in J$ , and  $f(x) \neq f(y)$  if  $x_j \neq y_j$  for some  $j \in J$  (where  $J \subseteq \{1, 2, \dots, k\}$  and  $|J| = k-1$ ).

**Examples.** Consider the Parity is  $n$ -sensitive everywhere. Majority is  $\lceil n/2 \rceil$ -sensitive everywhere. OR is only 1-sensitive everywhere. All zeros input is  $n$ -sensitive everywhere.

**2.2. Lower bounds for CRCW PRAM (1)**

**THEOREM 2.1.** Let  $M$  be a CRCW PRAM (1) that computes a  $k$ -sensitive function  $f$  in time  $T$  with  $m$  processors.

**COROLLARY 2.1.** Let  $M$  be a CRCW PRAM (1) that computes a  $k$ -sensitive function on  $n$  bits in time  $T$  with  $m$  processors.

**Proof.** Parity, sum and majority are  $k$ -sensitive everywhere.  $\square$

Let us informally discuss the proof. Consider the behavior of the PRAM on a  $k$ -sensitive function  $f$ .

**Case 1.** No processor writes in any time period.

**Case 2.** Some processor writes in some time period.

We have to analyze the information that is written in the memory cell  $C$ .

**Case 1.** As Cook and Dwork [CD-77] show, in this case, namely the information that is written in the memory cell  $C$  can be used in an algorithm for the function  $f$  in a way they keep track of this elusive information.

Read Exclusive-Write PRAM (CREW PRAM). Both models are precisely defined in § 2. In the above models processors communicate via a shared memory. Therefore the size of the communication facility of the machine, here called *communication width* (or *width* in short) is simply the number of shared memory cells. We consider the width  $m$  a resource, together with the size  $p$  (the number of processors) and the depth  $T$  (the running time), and we seek trade-offs between the three.

One of the subtleties in proving lower bounds for these models, is that information may be communicated by the fact that no processor writes into a common memory cell. We introduce a novel technique to deal with this difficulty.

For a large class of functions, which includes Parity and Majority, we prove  $T = \Omega((n/m)^{1/2})$  on the CRCW PRAM, where  $n$  is the size of the input. This lower bound is tight for all values of width  $m = O(n/\log^2 n)$ . This is the first time nontrivial tight lower bounds are achieved for a model that allows concurrent write access. The only known lower bound on the CRCW PRAM model is given in Stockmeyer and Vishkin [SV-82]. They show, using a result of Furst, Saxe, and Sipser [FSS-81], that it is impossible to compute parity in this model in constant time using a polynomial number of processors. There is, however, a large gap between this lower bound and the best upper bound known for a polynomial number of processors, which is  $O(\log n/\log \log n)$ . (See [CSV-82]).

For another class of functions, which includes the functions AND and OR, we prove a lower bound of  $T = \Omega((n/m)^{1/3})$  on the CREW PRAM. This lower bound extends the  $\Omega(\log n)$  of Cook and Dwork for small values of  $m$ , and further discerns the power of CRCW PRAM from the CREW PRAM. At this point we state, and give a proof, of a new result by Beame that achieves a tight lower bound for computing the OR in this model. For a different class of functions (that include OR) he proves  $T = \Omega((n/m)^{1/2})$ .

Both our lower bounds hold regardless of the number of processors, while the upper bounds are achieved with the smallest possible number of processors.

Our study of values of  $m$  which are smaller than input size requires us to add a read only input tape to the model, as is done in the study of space bounded Turing machines. The interest in those values is not solely theoretical—it is well founded in practice. For example the “Ethernet” can be considered as a PRAM with only one shared memory cell. Also, the papers Gottlieb et al. [GGKMRS-82], Kuck [K-77] and Vishkin [V-82] imply that minimizing the size of shared memory (that can be accessed in parallel) may amount to hardware feasibility of the parallel machine.

The paper is organized as follows: precise definitions and the lower bounds are given in § 2. Section 3 contains the upper bounds and § 4 concludes the paper and suggests further research directions. To improve the readability of § 2, some of the proofs were deferred to the appendix.

**2. Lower bounds.** In the first subsection we give precise definitions of the models of computation when the communication width  $m = 1$ , and of the types of functions we are interested in. Subsections 2.2 and 2.3 contain the lower bound proofs for the concurrent-write and exclusive-write models respectively when  $m = 1$ . In the last subsection we show how to extend the lower bounds for arbitrary communication width.

**2.1. Definitions.**

DEFINITION 2.1. A CRCW PRAM (1) consists of a set  $\Pi = \{p_1, p_2, \dots\}$  of processors, a number  $n$  of inputs,  $n$  read-only input cells  $X(1), X(2), \dots, X(n)$ , one common memory cell  $C$ , an alphabet  $\Sigma$  and an execution time  $T$ .

Each processor  $p_i$  has a  
 $\rho_i: Q_i \rightarrow \{1, 2, \dots, n\}$ -t  
 $\sigma_i: Q_i \rightarrow \Sigma$ -the symbol t  
 $\delta_i: Q_i \times \Sigma \times \Sigma \rightarrow Q_i$ -the s

At each time period  $t =$   
 cell  $C$  contains a symbol  $s^t$   
 $x_i \in \Sigma$  ( $1 \leq i \leq n$ ), the cell  $C$   
 $p_i$  is in an initial state  $q_i^0 \in Q_i$

$$q_i^{t+1} = \delta_i(q_i^t, s^t, x_{\rho_i})$$

$$s^{t+1} = \begin{cases} s^t & \text{if } \sigma_i(q_i^t) = s^t \\ \sigma_i(q_i^t) & \text{otherwise} \end{cases}$$

The value  $f(x_1, x_2, \dots,$   
 contents  $s^T$  of  $C$  at time  $T$ .

DEFINITION 2.2. A CR  
 (1), with only one exception-  
 that writes, i.e. at most one

Remarks. These lower b  
 the processors be nonuniform  
 Also note that we use the co  
 contents of  $C$ , (and in the C  
 doing so).

DEFINITION 2.3. Let  $\Sigma$   
 $x = x_1, x_2, \dots, x_n \in \mathbf{I}$  is said  
 $\{1, 2, \dots, n\}$ ,  $|J| = k - 1$  there  
 all  $j \in J$ , and  $f(x) \neq f(y)$ . If  $k$   
 $x \in \mathbf{I}$  is  $k$ -sensitive w.r.t.  $f$ , the  
 somewhere).

Examples. Consider the  
 Parity is  $n$ -sensitive every  
 Majority is  $\lceil n/2 \rceil$ -sensiti  
 OR is only 1-sensitive ev  
 all zeros input is  $n$ -sensitive v

**2.2. Lower bounds for C**

THEOREM 2.1. Let  $M$   
 everywhere function  $f$  in time

COROLLARY 2.1. Let  $M$   
 (Sum, Max) function on  $n$  bits

Proof. Parity, sum and ma  
 tive everywhere. □

Let us informally discuss t  
 2.1. Consider the behavior of th

Case 1. No processor wri  
 Case 2. Some processor (

We have to analyze the inf  
 Case 1. As Cook and Dwork

case, namely the information th  
 be used in an algorithm for the  
 way they keep track of this elu

Each processor  $p_i$  has a set of states  $Q_i$  and functions  
 $\rho_i: Q_i \rightarrow \{1, 2, \dots, n\}$ -the next input cell to be read,  
 $\sigma_i: Q_i \rightarrow \Sigma$ -the symbol to be written into  $C$ , and  
 $\delta_i: Q_i \times \Sigma \times \Sigma \rightarrow Q_i$ -the state transition function.

At each time period  $t = 0, 1, \dots, T$  each processor  $p_i$  is in a state  $q_i^t \in Q_i$ , and the cell  $C$  contains a symbol  $s^t \in \Sigma$ . At time  $t = 0$  the input cell  $X(i)$  contains the input  $x_i \in \Sigma$  ( $1 \leq i \leq n$ ), the cell  $C$  contains a designated symbol  $b_0 \in \Sigma$ , and every processor  $p_i$  is in an initial state  $q_i^0 \in Q_i$ . In general

$$q_i^{t+1} = \delta_i(q_i^t, X(j), s^t), \text{ where } j = \rho_i(q_i^t), \text{ and}$$

$$s^{t+1} = \begin{cases} s^t & \text{if for every } i \sigma_i(q_i^{t+1}) = s^t \text{ (no one writes),} \\ \sigma_i(q_i^{t+1}), & i \text{ is the smallest s.t. } \sigma_i(q_i^{t+1}) \neq s^t. \end{cases}$$

The value  $f(x_1, x_2, \dots, x_n)$  of the function  $f$  computed by the PRAM (1) is the contents  $s^T$  of  $C$  at time  $T$ .

DEFINITION 2.2. A CREW PRAM (1) is defined exactly like the CRCW PRAM (1), with only one exception—at each time period  $t$  there can be at most one processor that writes, i.e. at most one  $i$  s.t.  $\sigma_i(q_i^{t+1}) \neq s^t$ .

Remarks. These lower bound models allow  $\Pi$ ,  $\Sigma$  and  $Q_i$  to be infinite, and allow the processors be nonuniform (i.e. have different programs for different values of  $n$ ). Also note that we use the convention that a processor writes if it tries to change the contents of  $C$ , (and in the CRCW it must be the one with the smallest serial number doing so).

DEFINITION 2.3. Let  $\Sigma$  be a set,  $\mathbf{I} \subseteq \Sigma^n$  and  $f: \mathbf{I} \rightarrow \Sigma$  some function. An input  $x = x_1, x_2, \dots, x_n \in \mathbf{I}$  is said to be  $k$ -sensitive w.r.t.  $f$  if for every subset  $J \subseteq \{1, 2, \dots, n\}$ ,  $|J| = k - 1$  there exists another input  $y = y_1, y_2, \dots, y_n \in \mathbf{I}$  s.t.  $x_j = y_j$  for all  $j \in J$ , and  $f(x) \neq f(y)$ . If  $k$  is the largest integer s.t. every input (resp. some input)  $x \in \mathbf{I}$  is  $k$ -sensitive w.r.t.  $f$ , then  $f$  is said to be  $k$ -sensitive everywhere (resp.  $k$ -sensitive somewhere).

Examples. Consider the functions Parity, Majority,  $\text{OR}: \{0, 1\}^n \rightarrow \{0, 1\}$ .

Parity is  $n$ -sensitive everywhere.

Majority is  $\lceil n/2 \rceil$ -sensitive everywhere.

OR is only 1-sensitive everywhere for all  $n$ , but it is  $n$ -sensitive somewhere (the all zeros input is  $n$ -sensitive w.r.t. OR).

**2.2. Lower bounds for CRCW PRAM (1).**

THEOREM 2.1. Let  $M$  be a CRCW PRAM (1) that computes a  $k$ -sensitive everywhere function  $f$  in time  $T$ . Then  $T = \Omega(\sqrt{k})$ .

COROLLARY 2.1. Let  $M$  be a CRCW PRAM (1) that computes the Parity, Majority (Sum, Max) function on  $n$  bits (integers) in time  $T$ . Then  $T = \Omega(\sqrt{n})$ .

Proof. Parity, sum and max are  $n$ -sensitive everywhere. Majority is  $\lceil n/2 \rceil$ -sensitive everywhere.  $\square$

Let us informally discuss the difficulties we are facing in trying to prove Theorem 2.1. Consider the behavior of the machine in time period  $t$ . There are two possible cases:

Case 1. No processor writes into  $C$  ( $s^t = s^{t-1}$ .)

Case 2. Some processor (say  $p_i$ ) writes into  $C$  ( $s^t \neq s^{t-1}$ ).

We have to analyze the information that is transferred in each case. Consider first Case 1. As Cook and Dwork [CD-82] point out, information is transferred in this case, namely the information that nobody wrote. They show how this information can be used in an algorithm for the OR function, that is faster than the obvious one. The way they keep track of this elusive information is heavily based on the fact that their

models are precisely defined in shared memory. Therefore the called communication width memory cells. We consider the of processors) and the depth e three.  
 e models, is that information es into a common memory difficulty.

ty and Majority, we prove size of the input. This lower is the first time nontrivial concurrent write access. The is given in Stockmeyer and e, and Sipser [FSS-81], that ant time using a polynomial tween this lower bound and er of processors, which is

unctions AND and OR, we PRAM. This lower bound s of  $m$ , and further discerns this point we state, and give lower bound for computing that include OR) he proves

per of processors, while the mber of processors.

out size requires us to add a dy of space bounded Turing oretical—it is well founded ered as a PRAM with only al. [GGKMRS-82], Kuck e size of shared memory are feasibility of the parallel

s and the lower bounds are 4 concludes the paper and ability of § 2, some of the

ise definitions of the models nd of the types of functions lower bound proofs for the ly when  $m = 1$ . In the last itrary communication width.

set  $\Pi = \{p_1, p_2, \dots\}$  of pro-  $X(1), X(2), \dots, X(n)$ , one time  $T$ .

model does not allow simultaneous write access to the same memory cell. (Indeed, their lower bound does not hold for the CRCW PRAM). As our model allows simultaneous write access, we had to choose an approach which is different from theirs.

The information that is transferred in Case 2 seems even more slippery. We know what was written into  $C$ , and in addition we know that no processor with serial number smaller than  $j$  tried to write. (Note that as  $\Sigma$  may be infinite, the writer can encode its serial number in the symbol it writes.) This case is much simpler in the exclusive write model, since there, if someone writes, there can be no other processor that tries to write!

At this point we need some notation. Let  $\mathbf{I}$  denote the (nonempty) set of all possible inputs (the domain). Fix a time period  $t$  and let  $\beta = s^0 s^1 \dots s^{t-1}$  be the string of successive symbols in  $C$  in time periods  $0, 1, \dots, t-1$ .  $\beta$  is called the history through time  $t$ . Denote by  $\mathbf{I}_\beta \subseteq \mathbf{I}$  the subset of inputs that have history  $\beta$  through time  $t$ .

Our analysis will be based on the observation that Cases 1 and 2 consist each of two subcases. Fix  $\beta$ , a history through time  $t$ .

Case 1a. There is no input in  $\mathbf{I}_\beta$  for which some processor writes at time  $t$ .

Case 1b. There is an input in  $\mathbf{I}_\beta$  for which some processor writes at time  $t$ .

Case 2a. There is no input in  $\mathbf{I}_\beta$  for which some processor with smaller serial number than  $j$  writes at time  $t$ .

Case 2b. There is an input in  $\mathbf{I}_\beta$  for which a processor with smaller serial number than  $j$  writes at time  $t$ .

It turns out that Cases 1a and 2a are simple to analyze. Intuitively, in Case 1a no new information is transferred as  $\beta$  itself contains the information that no one will write at time  $t$ . Similarly, in Case 2a,  $\beta$  contains the information that no processor with a smaller serial number than the writer could have written, so the only new piece of information is the new symbol in  $C$ ,  $s^t$ .

Now, rather than confronting the elusive information that is transferred in Cases 1b and 2b, we avoid (or circumvent) it, and hence coin the name *circumvention* for this technique. Showing that we can restrict ourselves to the "easy to analyze" cases is the heart of our argument.

Let  $\mathbf{I}(\{(i_1, y_1), \dots, (i_l, y_l)\}) = \{x \in \mathbf{I} \mid x_{i_j} = y_j, 1 \leq j \leq l\}$  denote the set of all inputs ( $n$ -tuples) whose projection on the  $l$ -tuple  $(i_1, i_2, \dots, i_l)$  is  $(y_1, y_2, \dots, y_l)$ .

*Remark.* We switch here from qualifying inputs by their history ("range" qualification) to qualifying them by their values at given coordinates (domain qualification). This yields a simpler and more intuitive proof than our original one which used range qualification. However, we believe that range qualification is more powerful, and that it may be used to prove lower bounds when domain qualification fails.

The following iterative definition will generate an "easy to analyze" set of inputs, i.e. inputs for which Cases 1b and 2b never occur. For every  $t$ ,  $D^t$  will contain pairs of "fixed" input positions and their values, and  $E^t = \mathbf{I}(D^t)$ .

Let  $E^0 = \mathbf{I}$  and  $D^0 = \emptyset$ . Consider time period  $t$  and define  $E^t, D^t$  according to the following:

Case 1. There is no processor  $p_j$  and no input  $x \in E^{t-1}$  such that  $p_j$  writes on  $x$  at time  $t$ . Then

$$E^t \leftarrow E^{t-1}, \quad D^t \leftarrow D^{t-1}.$$

Case 2. There is a processor  $p_j$  and an input  $x \in E^{t-1}$  s.t.  $p_j$  writes on  $x$  at time  $t$ . Let  $p_l$  and  $y \in E^{t-1}$  be so that  $p_l$  writes on  $y$  at time  $t$ , and  $l$  is the smallest serial number of any processor that writes at time  $t$  on any input in  $E^{t-1}$ . Let  $i_1, i_2, \dots, i_u$  and  $y_{i_1}, y_{i_2}, \dots, y_{i_u}$  be the sets of input cells and their contents (respectively) that were

read by  $p_l$  up to time  $t$ . (

It is easy to see that

(1)  $|D^t| \leq |D^{t-1}| + t$ ,

(2)  $E^t = \mathbf{I}(D^t)$ ,  $D^{t-1}$

(3)  $E^t \neq \emptyset$ .

In particular we have

LEMMA 2.1.  $E^T \neq \emptyset$

*Remark.* The definiti regardless of the function can be used to prove lower

LEMMA 2.2. Let  $M$  be defined as above for  $M$ .

A rigorous proof of inductively on  $t$ , that any have exactly the same com

*Proof of Theorem 2.1.*

$f$  in time  $T$ . Suppose that there must be inputs  $x$  and Therefore  $T(T+1)/2 \geq k$ ,

### 2.3. Lower bounds fo

bits. As mentioned earlier, previous subsection impl CRCW PRAM (1). Indeed follows. In the first step, the second step, a processor  $p_i$  value it read was "1."

It is clear why this alg that if the domain consists a "1," a write conflict cannot For this reason we will res  $\mathbf{I} = \Sigma^n$ ). The main result in

THEOREM 2.2. Let  $N$  where function  $g$  in time  $T$ .

COROLLARY 2.2. If  $g$

In an earlier version of the 2.2 can be improved to  $T$  fact, he proved the following

THEOREM 2.3. (Beame  $g: \{0, 1\}^n \rightarrow \{0, 1\}$  in time  $T$ .

then  $T = \Omega(\sqrt{\log_2 r})$ .

It immediately follows

COROLLARY 2.3 (Beame

The proof of Theorem 2 while we focus on the sensitivity on a different parameter, nar is of independent interest, a

read by  $p_t$  up to time  $t$ . (Clearly  $u \leq t$ .) Then

$$E^t \leftarrow E^{t-1} \cap \mathbf{I}(\{(i_1, y_{i_1}), \dots, (i_u, y_{i_u})\}),$$

$$D^t \leftarrow D^{t-1} \cup \{(i_1, y_{i_1}), \dots, (i_u, y_{i_u})\}.$$

It is easy to see that for every  $0 \leq t \leq T$

(1)  $|D^t| \leq |D^{t-1}| + t, |D^0| = 0$  and hence  $|D^t| \leq t(t+1)/2$ .

(2)  $E^t = \mathbf{I}(D^t), D^{t-1} \subseteq D^t, E^t \subseteq E^{t-1}$ .

(3)  $E^t \neq \emptyset$ .

In particular we have:

LEMMA 2.1.  $E^T \neq \emptyset$  and  $|D^T| \leq T(T+1)/2$ .

*Remark.* The definition above generates a set  $E^T$  of "easy to analyze" inputs, regardless of the function being computed. Therefore we believe that this technique can be used to prove lower bounds for the computation of other functions in this model.

LEMMA 2.2. Let  $M$  be an CRCW PRAM (1) computing a function  $f$ , and let  $E^T$  be defined as above for  $M$ . Then for every  $x, y \in E^T, f(x) = f(y)$ .

A rigorous proof of this lemma is given in the appendix. The idea is to show inductively on  $t$ , that any processor which writes at time  $t$  on some input in  $E^T$ , will have exactly the same computation through time  $t$  on every input in  $E^T$ .

*Proof of Theorem 2.1.* Recall that  $M$  computes a  $k$ -sensitive everywhere function  $f$  in time  $T$ . Suppose that  $T(T+1)/2 < k$ . Then  $|D^T| > k$ , and so by Definition 2.3, there must be inputs  $x$  and  $y$  in  $E^T$  s.t.  $f(x) \neq f(y)$ . This contradicts Lemma 2.2. Therefore  $T(T+1)/2 \geq k$ , so  $T = \Omega(\sqrt{k})$ .  $\square$

**2.3. Lower bounds for the CREW PRAM (1).** Consider the OR function of  $n$  bits. As mentioned earlier, the OR is just 1-sensitive everywhere, so the results in the previous subsection imply only a constant time lower bound for it on the CRCW PRAM (1). Indeed, there is a two step algorithm for the OR on this model as follows. In the first step, the common memory cell  $C$  is initialized with "0." In the second step, a processor  $p_i$  reads the  $i$ th input position and writes a "1" into  $C$  iff the value it read was "1."

It is clear why this algorithm is not valid for a CREW PRAM. Note, however, that if the domain consists only of inputs which have at most one position containing a "1," a write conflict cannot occur, and the algorithm is valid for the CREW PRAM. For this reason we will restrict ourselves here to functions with a full domain (i.e.  $\mathbf{I} = \Sigma^n$ ). The main result in this subsection is the following theorem.

THEOREM 2.2. Let  $N$  be a CREW PRAM (1) that computes a  $k$ -sensitive somewhere function  $g$  in time  $T$ . Then  $T = \Omega(k^{1/3})$ .

COROLLARY 2.2. If  $g$  is the OR function on  $n$  bits, then  $T = \Omega(n^{1/3})$ .

In an earlier version of this paper we conjectured that the lower bound of Corollary 2.2 can be improved to  $T = \Omega(\sqrt{n})$ . This was recently proved by Beame [B-83]. In fact, he proved the following stronger theorem.

THEOREM 2.3. (Beame). Let  $N$  be a CREW PRAM (1) that computes a function  $g: \{0, 1\}^n \rightarrow \{0, 1\}$  in time  $T$ . If there exists an input  $e \in \mathbf{I}$  s.t.  $|\{x \in \mathbf{I}: g(x) = g(e)\}| \leq |\mathbf{I}|/r$ , then  $T = \Omega(\sqrt{\log_2 r})$ .

It immediately follows that:

COROLLARY 2.3 (Beame). If  $g$  is the OR function on  $n$  bits, then  $T = \Omega(\sqrt{n})$ .

The proof of Theorem 2.3 has the same structure as that of Theorem 2.2. However, while we focus on the sensitivity of inputs in the lower bound argument, Beame focuses on a different parameter, namely the number of inputs with the same image. His proof is of independent interest, and we include it in the appendix.

We return to the proof of Theorem 2.2. The idea is to use the framework of the previous subsection, namely to construct a set of inputs  $E^T$ , and show that for the computed function to be constant on  $E^T$ ,  $T$  must be large. This task was relatively easy for everywhere sensitive functions, since we did not have to worry about the contents of  $E^T$ , as every input is sensitive. To use the sensitivity of inputs in a somewhere sensitive function in a similar argument we must make sure that  $E^T$  contains at least one sensitive input. This motivates the following inductive definition of the sets  $D^t, E^t$ .

Let  $g: \mathbf{I} \rightarrow \Sigma$  be the function being computed and  $e = e_1, e_2, \dots, e_n \in \mathbf{I}$  be a  $k$ -sensitive input w.r.t.  $g$ . Set  $E^0 = \mathbf{I}$  and  $D^0 = \emptyset$ . Consider time period  $t$  and define  $E^t, D^t$  as follows:

Case 1. There is no processor  $p_j$  and no input  $x \in E^{t-1}$  such that  $p_j$  writes on  $x$  at time  $t$ . Then

$$E^t \leftarrow E^{t-1}, \quad D^t \leftarrow D^{t-1}.$$

Case 2. There is a (unique) processor  $p_j$  that writes on  $e \in E^{t-1}$  at time  $t$ . Let  $i_1, i_2, \dots, i_u$  and  $e_{i_1}, e_{i_2}, \dots, e_{i_u}$  be the sets of input cells and their contents (respectively) that were read by  $p_j$  up to time  $t$ . (Clearly  $u \leq t$ ). Then

$$D^t \leftarrow D^{t-1} \cup \{(i_1, e_{i_1}), \dots, (i_u, e_{i_u})\},$$

$$E^t \leftarrow E^{t-1} \cap \mathbf{I}(\{(i_1, e_{i_1}), \dots, (i_u, e_{i_u})\}).$$

Case 3. There exists  $x \in E^{t-1}, x \neq e$  s.t. some  $p_j$  writes on  $x$  at time  $t$ , but no processor writes on  $e$  at time  $t$ . Let  $R_0^t$  be a set of positions s.t. if  $y \in E^{t-1}$  and  $y_i = e_i$  for all  $i \in R_0^t$ , then no processor writes on  $y$  at time  $t$ . In this case we fix the positions  $R_0^t$  with values of  $e$ :

$$D^t \leftarrow D^{t-1} \cup \{(i, e_i) | i \in R_0^t\},$$

$$E^t \leftarrow E^{t-1} \cap \mathbf{I}(\{(i, e_i) | i \in R_0^t\}).$$

It is easy to see inductively that  $e \in E^t$  for all  $t$ , and so  $e \in E^t$ . Our main problem is to obtain an upper bound on  $|R_0^t|$ .

LEMMA 2.3. For every  $t, |R_0^t| \leq t(t+1)/2$ .

This lemma is the heart of the lower bound. Since the proof is long, it is deferred to the appendix.

LEMMA 2.4. For every  $t, |D^t| \leq t(t+1)(t+2)/6$  and  $e \in E^t$ .

Proof. By simple induction on  $t$ .  $\square$

LEMMA 2.5. For every  $x, y \in E^T, g(x) = g(y)$ .

Proof. Exactly the same as the proof of Lemma 2.2.  $\square$

Proof of Theorem 2.2. Recall that  $N$  computes a  $k$ -sensitive somewhere function  $g$  in time  $T$ . Suppose  $T(T+1)(T+2)/6 < k$ . Then by Lemma 2.4  $|D^T| > k$ . Since  $e \in E^T$ , by Definition 2.3 there must be a  $y \in E^T$  s.t.  $g(y) \neq g(e)$ , which contradicts Lemma 2.5.  $\square$

**2.4. Arbitrary communication width.** What happens when the communication width is larger than 1? The CRCW PRAM( $m$ ) is defined similarly to the CRCW PRAM(1), only now there are  $m$  common memory cells  $C(1), C(2), \dots, C(m)$  to which the processors have concurrent read/write access. In a similar fashion the CREW PRAM( $m$ ) can be defined. Our results are summarized in the following theorem.

THEOREM 2.4. Let  $M$  be a CRCW PRAM( $m$ ) that computes a  $k$ -sensitive everywhere function  $f: \mathbf{I}(\subseteq \Sigma^n) \rightarrow \Sigma$  in time  $T$ . Then  $T = \Omega(\sqrt{k/m})$ . In particular, if  $f \in \{\text{Parity, Majority, Sum, Max}\}, T = \Omega(\sqrt{n/m})$ . Let  $N$  be a CREW PRAM( $m$ ) that

computes a  $k$ -sensitive somewhere function  $f$  in time  $T$  if  $g \in \{\text{AND, OR}\}, T = \Omega(\sqrt{k/m})$ .

The only difficulty in the definition of the "easy to compute" functions is for which the following holds: for any input  $x$  written into. However, if  $v$  writes into  $C(1)$ , no one else writes into  $C(1)$ .

We overcome this by using different cells as follows: only cell  $C(i)$  may be written into not only to the contents of  $C(i)$  but also to the contents of cells 1 to  $i-1$  at time  $t$ . This is done by conceptual slicing. Indeed, we allow the processors to write into  $C(i)$  only if the result we are able to define in this fashion to the previous PRAM( $m$ ) respectively. Then we define  $E^t$  from  $E^{t-1}$  and  $D^t$  from  $D^{t-1}$ .

The analysis of the lower bound is done in the same manner w.r.t the final sets  $E^t, D^t$ . The lemmas and the conclusion are similar to the previous ones.

LEMMA 2.6. In the CREW PRAM( $m$ ) model,

In the CREW PRAM( $m$ ) model,

LEMMA 2.7. In the CREW PRAM( $m$ ) model,

In the CREW PRAM( $m$ ) model,

We conclude this subsection with two ideas on how to improve the upper bound (Theorem 2.3) for arbitrary functions.

THEOREM 2.5. Let  $N$  be a CREW PRAM( $m$ ) that computes a function  $f: \{0, 1\}^n \rightarrow \Sigma$  in time  $T$ . If there exists a constant  $c$  such that  $T = \Omega(\sqrt{(\log_2 r)/m})$ . In particular,

(1) Two other concurrent read/write models are known in the literature ([SV-81], [S81]).

(2) Two other concurrent read/write models are known in the literature ([SV-81], [S81]).

they resolve write conflicts. The location should write the value. We do not know in advance which location should write the value. This section by mentioning that the results for the CRCW PRAM( $m$ ) model.

we do not know in advance which location should write the value.

section by mentioning that the results for the CRCW PRAM( $m$ ) model.

**3. Upper bounds. All-processor CREW PRAM.**

PRAM, namely the Exclusive-Read CREW PRAM. A shared memory cell is for all processors to write into informally. They will be given a location to see that they hold for all processors.

Consider first the CREW PRAM( $m$ ) model. Initially stored in the read-only memory  $p_j$ , and  $C$  is the common memory cell.

Clearly, only  $p = O(\sqrt{n})$  processors can be used. This is computed in  $O(\sqrt{n})$  time.

computes a  $k$ -sensitive somewhere function  $g: \Sigma^n \rightarrow \Sigma$ . Then  $T = \Omega((k/m)^{1/3})$ . In particular, if  $g \in \{\text{AND, OR}\}$ ,  $T = \Omega((n/m)^{1/3})$ .

The only difficulty in extending our technique to prove Theorem 2.4 is in the definition of the "easy to analyze" cases. For example, one can construct a machine for which the following happens: There are inputs for which both  $C(1)$  and  $C(2)$  are written into. However, if we choose an input for which the smallest numbered processor writes into  $C(1)$ , no one will write into  $C(2)$  and vice versa.

We overcome this difficulty by conceptually serializing the write access into different cells as follows: Each time unit  $t$  is sliced into  $m$  slices, so that in the  $i$ th slice only cell  $C(i)$  may be written into. Then, at the  $i$ th slice of time period  $t$  we can refer not only to the contents of all cells at previous time periods, but also to the contents of cells 1 to  $i-1$  at time period  $t$ . (Note that the machine is not affected by this conceptual slicing. Indeed, it shows that our results hold even in a stronger model that allows the processors to access all common memory cells at each time unit.) As a result we are able to define sets  $E^t$  and  $D^t$ ,  $0 \leq t \leq T$ ,  $1 \leq i \leq m$ , inductively in a similar fashion to the previous subsections for the CRCW PRAM( $m$ ) and the CREW PRAM( $m$ ) respectively. The only refinement is that instead of defining  $E^t$  from  $E^{t-1}$ , we define  $E^t$  from  $E^{t-1}$  when  $i > 1$ , and  $E^t$  from  $E^{(t-1)m}$ .

The analysis of the previous subsections carries through in a straightforward manner w.r.t the final sets,  $E^{Tm}$  and  $D^{Tm}$ . This includes the proof of the following two lemmas and the conclusion of the theorem from them.

LEMMA 2.6. In the CRCW PRAM( $m$ ),  $|D^{Tm}| \leq m(T+1)T/2$ .

In the CREW PRAM( $m$ ),  $|D^{Tm}| \leq mT(T+1)(T+2)/6$ .

LEMMA 2.7. In the CRCW PRAM( $m$ ), for every  $x, y \in E^{Tm}$ ,  $f(x) = f(y)$ .

In the CREW PRAM( $m$ ), for every  $x, y \in E^{Tm}$ ,  $g(x) = g(y)$ .

We conclude this subsection with two observations:

(1) The ideas outlined above can be used to extend also Beame's theorem (Theorem 2.3) for arbitrary communication width, as follows.

THEOREM 2.5. Let  $N$  be a CRCW PRAM( $m$ ) that computes a function  $g: \{0, 1\}^n \rightarrow \{0, 1\}$  in time  $T$ . If there exists an input  $e \in I$  s.t.  $|\{x \in I: g(x) = g(e)\}| \leq |I|/r$ , then  $T = \Omega(\sqrt{(\log_2 r)/m})$ . In particular, if  $g$  is the OR function, then  $T = \Omega(\sqrt{n/m})$ .

(2) Two other concurrent-write models of parallel computation that appeared in the literature ([SV-81], [ShV-82]). They differ from our CRCW PRAM in the way they resolve write conflicts. In the first all processors that access the same memory location should write the same value. In the second there is no such restriction, but we do not know in advance which processor succeeds in writing. We conclude this section by mentioning that those two models are weaker than ours, and therefore our results for the CRCW PRAM hold for them as well.

**3. Upper bounds.** All upper bounds can be achieved in the weakest version of a PRAM, namely the Exclusive-Read Exclusive-Write PRAM (EREW PRAM). It is similar to the CREW PRAM, only that in this model any simultaneous access of a shared memory cell is forbidden. The algorithms are simple and will be described informally. They will be given only for the problem of summing  $n$  numbers. It is easy to see that they hold for computing any associative function.

Consider first the EREW PRAM (1) model. The  $n$  numbers  $a_1, a_2, \dots, a_n$  are initially stored in the read-only input tape. Let  $L_i$  be a local memory cell of processor  $p_i$ , and  $C$  is the common memory cell. The algorithm is described in Fig. 1.

Clearly, only  $p = O(\sqrt{n})$  processors are active in this algorithm, and the sum is computed in  $O(\sqrt{n})$  time. Since sequential time for summation is  $\Omega(n)$ , a

| Time | $p_1$                                      | $p_2$  | $p_3$  | $p_4$   | ... |
|------|--|--|--|---|-----|
| 1    | $L_1 \leftarrow a_1$<br>$C \leftarrow L_1$ | $L_2 \leftarrow a_2$                                 | $L_3 \leftarrow a_4$                                 | $L_4 \leftarrow a_7$                                    | ... |
| 2    |  | $L_2 \leftarrow L_2 + a_3$<br>$C \leftarrow C + L_2$ | $L_3 \leftarrow L_3 + a_5$                           | $L_4 \leftarrow L_4 + a_8$                              | ... |
| 3    |  |  | $L_3 \leftarrow L_3 + a_6$<br>$C \leftarrow C + L_3$ | $L_4 \leftarrow L_4 + a_9$                              | ... |
| 4    |  |  |  | $L_4 \leftarrow L_4 + a_{10}$<br>$C \leftarrow C + L_4$ | ... |

FIG. 1. Summation with one common memory cell.

straightforward lower bound of  $\Omega(n/p)$  exists for any parallel machine with  $p$  processors. Hence the number of processors is optimal up to a constant factor.

Consider now the same problem for the CRCW PRAM( $m$ ), where  $m = O(n/\log^2 n)$ . We show how to achieve  $O(\sqrt{n/m})$  time with  $O(\sqrt{nm})$  processors. The algorithm has two phases:

(1) Partition the  $n$  inputs into  $m$  subsets of size roughly  $n/m$  each. Assign to each subset  $\sqrt{n/m}$  processors and one common memory cell. For each subset the sum is computed in the respective memory cell using the algorithm above in time  $O(\sqrt{n/m})$ .

(2) Sum up the  $m$  values in the common memory using  $m(\leq \sqrt{nm})$  processors in  $O(\log m)$  time in the obvious way.

As before, the number of processors used is optimal up to a constant factor. This upper bound establishes that our lower bound for Parity on the CRCW PRAM( $m$ ) and Beame's lower bound for the OR on the CREW PRAM( $m$ ) are tight.

We conclude by mentioning what is known when the communication width is larger than the input size. If the input values are taken from a finite domain, the sum can be computed in constant time using exponential width and number of processors. If those two resources are bounded by a polynomial in  $n$ , the best upper bound known is  $O(\log n/\log \log n)$  [CSV-82].

**4. Conclusions and open problems.** Using communication based arguments to prove lower bounds in computer science is an old idea. The crossing-sequence [HU-79] technique in Turing machines essentially measures communication between work-tape cells. This technique was extended to measure communication between two halves of a VLSI circuit [Y-81], [LS-81], [PS-82] and obtain Time-Area trade-offs.

We consider this paper to be a first step towards understanding the central role played by communication in efficient parallel computation. The view of communication as a resource in parallel machines gives rise to many questions. We mention a few below.

(1) Our lower bound for the OR on the CREW PRAM, combined with that of Cook and Dwork, covers the whole range of  $m$ . On the other hand, the lower bound for the parity functions on the CRCW PRAM( $m$ ) becomes trivial when  $m \geq n$ . The case where  $m$  is only bounded by a polynomial in  $n$  is of particular interest, since a lower bound on the time here will give a lower bound on the depth of polynomial size parity circuits.

(2) Consider para or nondeterministic. In studied here, both the the time. If we allow no in constant time. Howe even in the nondetermi

(3) Study Time-W

**Appendix.**

LEMMA 2.2. For et Proof of Lemma 2.

- $q'_i(x)$  and  $s^i(x)$  are the input  $x$ .
- $R'_j(x) = \{\rho_j(q'_j(x))\} \cup R'_0(x) = \emptyset$ .
- $w^i(x)$  is the index of such processor at time  $t$ .
- $W^i(x) = \bigcup_{r=1}^i R'_r(x)$  through time  $t$ .
- $FW^i(x) = \{w^r(x) | t \leq r \leq i\}$  on.

Let  $x$  and  $y$  be elements. We prove by induction on  $t$  that for every  $j \in FW^i(x)$ ,  $q'_j(y) = q'_j(x)$  for  $t = 0$ . For every processor  $j$ ,  $s^0(y) = b_0$ ,  $W^0(x) = W^0(y)$ .

$t > 0$ . Assume the claim holds for  $t-1$ .  $i(y) = \rho_j(q'_j(y))$ . By the induction hypothesis, since  $j \in FW^i(x)$ ,  $R'_j(x) \subseteq W^i(x)$  and the induction hypothesis,  $\sigma_j(y) = \sigma_j(q'_j(y))$ . Then  $\sigma_j(y) = \sigma_j(q'_j(y))$ .

Case 1.  $s^t(x) = s^{t-1}(x)$ .  $s^{t-1}(x)$ ,  $w^t(x) = w^t(y) = 0$ .

Case 2.  $s^t(x) \neq s^{t-1}(x)$ . There is no  $l < j$  s.t.  $\sigma_l(q'_l(y)) \neq s^{t-1}(x)$ .  $s^t(y) = s^t(x)$ , and  $W^t(y) = W^t(x)$ .

LEMMA 2.3. For every  $S$ , denote only positive integers.

- $x \equiv y \pmod S$  means  $x - y \in S$ .
- $\mathbf{I}_y(S) = \{x \in \mathbf{I} | x \equiv y \pmod S\}$ .

Claim. Given an integer  $t$ , sets  $S_j \subseteq \{1, 2, \dots, n\}$  for  $j = 1, \dots, t$ .

- (1)  $S_j \cap S = \emptyset$  for all  $j$ .
- (2)  $|S_j| \leq t$  for all  $j$ .
- (3)  $h(e) = 0$ .
- (4)  $h(x) = j$  and  $y \equiv x \pmod S$ .
- (5)  $h(x) = j$ ,  $h(y) = k$ .

Then there exists a set  $R \subseteq \mathbf{I}$  of size  $|R| \geq n/2$ . Connection between the existence of  $t(t+1)/2$  inputs.



no one writes at time  $t$  in Case 3. Let  $R'_j$  be defined as in the proof of Lemma 2.2. Then let  $S$  be the set of fixed input positions through time  $t$ , ( $S = D^{t-1}$ ,  $\mathbf{I}_e(S) = E^{t-1}$ ),  $S_j = R'_j - S$  for all  $j$ , and let the function  $h: \mathbf{I}_e(S) \rightarrow Z^+$  be defined by  $h(x) = j$  if  $p_j$  is the (unique) processor that writes on  $x$  at time  $t$ , and  $h(x) = 0$  if no one writes on  $x$  at time  $t$ . Let us verify that properties (1)–(5) hold.

- (1) By the definition of  $S_j$ .
- (2)  $|S_j| = |R'_j - S| \leq |R'_j| \leq t$ .
- (3) We deal here only with case 3, in which no processor writes on  $e$ .
- (4) Since  $x, y \in \mathbf{I}_e(S)$  and  $x \equiv y \pmod{S_j}$ ,  $x \equiv y \pmod{R'_j}$ . With an almost identical proof to that of Lemma 2.2 we can prove that  $q'_j(x) = q'_j(y)$ , i.e.  $p_j$  will arrive at the same state at time  $t$  for both inputs  $x$  and  $y$ . In particular,  $p_j$  will write on  $x$  if and only if it will write on  $y$  at time  $t$ .
- (5) Suppose not. Then define an input  $z$  by  $z_i = x_i$  if  $i \in S_j$ ,  $z_i = y_i$  if  $i \in S_k - S_j$  and  $z_i = e_i$  for the remaining values of  $i$ . Clearly  $z \in \mathbf{I}_e(S) = E^{t-1}$ . Therefore both  $p_j$  and  $p_k$  write at time  $t$  on  $z$ , contradicting the definition of the CREW PRAM.

Now we can take  $R'_0 = R$ , which completes the proof of the lemma.  $\square$

*Proof of Claim.* The proof is by induction on  $t$ .

$t = 0$ . In this case  $h(\mathbf{I}_e(S)) = \{0\}$ . Otherwise, for some  $x \in \mathbf{I}_e(S)$  there exists  $j > 0$  s.t.  $h(x) = j$ , then (4) also  $h(e) = j$ , contradiction to (3).

$t > 0$ . If  $h(\mathbf{I}_e(S)) = \{0\}$  we are done. Assume that for some  $x \in \mathbf{I}_e(S)$ ,  $h(x) = l > 0$ . Set  $S' = S \cup S_l$ , and  $h'$  be the restriction of  $h$  to  $\mathbf{I}_e(S')$ , and  $S'_j = S_j - S_l$  for all  $j \in h'(\mathbf{I}_e(S'))$ . Then we have the following:

- (1')  $S' \cap S'_j = \emptyset$  for all  $j$ . Clear.
- (2')  $|S'_j| \leq t - 1$  for all  $j \in h'(\mathbf{I}_e(S'))$ . Since  $l, j \in h(\mathbf{I}_e(S))$ , by (5)  $S_j \cap S_l \neq \emptyset$ , and therefore  $|S'_j| = |S_j - S_l| \leq |S_j| - 1 \leq t - 1$ .
- (3')  $e \in \mathbf{I}_e(S')$  and  $h'(e) = 0$ . Clear.
- (4')  $h'(x) = j$ ,  $y \equiv x \pmod{S'_j}$  implies  $h'(y) = j$ . Since  $x, y \in \mathbf{I}_e(S')$ ,  $y \equiv x \equiv e \pmod{S_l}$  and therefore  $y \equiv x \pmod{S_j}$ . Hence  $j = h'(x) = h(x) = h(y) = h'(y)$ .
- (5')  $h'(x) = j$ ,  $h'(y) = k$ ,  $j \neq k$  implies that there is an  $i \in S'_j \cap S'_k$  s.t.  $x_i \neq y_i$ . Since  $h(x) = j$ ,  $h(y) = k$  there must be such an  $i$  in  $S_j \cap S_k$ . However, since  $x \equiv y \equiv e \pmod{S_l}$   $i$  must belong to  $S'_j \cap S'_k$ .

By the induction hypothesis, there exists a set  $R'$  s.t.  $|R'| \leq (t-1)t/2$  and  $h'(\mathbf{I}_e(S' \cup R')) = 0$ . Set  $R = R' \cup S_l$ . Then clearly  $|R| \leq t(t+1)/2$  and  $h(\mathbf{I}_e(S \cup R)) = h(\mathbf{I}_e(S' \cup R')) = h'(\mathbf{I}_e(S' \cup R')) = 0$ .  $\square$

**THEOREM 2.3.** *Let  $N$  be a CREW PRAM (1) that computes a function  $g$  in time  $T$  such that  $\exists e \in \mathbf{I}$  ( $\mathbf{I} = \{0, 1\}^n$ ) for which  $|\{x \in \mathbf{I} | g(x) \neq g(e)\}| \leq |\mathbf{I}|/r$ . Then  $T = \Omega\sqrt{\log_2 r}$ .*

Set  $E^0 = \mathbf{I}$  and  $F^0 = \mathbf{I} - E^0 = \emptyset$ . Consider time period  $t$ . For any  $j$  let  $P'_j$  be the set of input positions read by processor  $p_j$  up to time  $t$  and define  $E^t$  and its complement  $F^t$  as follows:

*Case 1.* There is no processor  $p_j$  and no input  $x \in E^{t-1}$  such that  $p_j$  writes on  $x$  at time  $t$ : Then  $E^t \leftarrow E^{t-1}$ ,  $F^t \leftarrow F^{t-1}$ .

*Case 2.* No processor writes on  $e$  at time  $t$  but there is an  $x \in E^{t-1}$  such that some  $p_j$  writes on  $x$  at time  $t$ : For every input  $x \in E^{t-1}$  which causes a processor  $p_j$  to write at time  $t$  define

$$C'_x = \mathbf{I}(\{(i, x_i) | i \in P'_j\}).$$

Each  $C'_x$  is specified by the values in at most  $t$  input cells since  $|P'_j| \leq t$ . It is clear that any  $x \in E^{t-1}$  which causes a write at time  $t$  is in some  $C'_y$ . Also any  $y \in E^{t-1} \cap C'_x$  will cause a write at time  $t$  since processor  $p_j$  at time  $t$  will not be able to distinguish  $y$  from  $x$ . Thus if we eliminate the elements of these "cubes" from  $E^{t-1}$  no writes will occur at time  $t$ .

The "cubes" also specify positions must be specified by different  $i$  then that  $C'_x \cap C'_y \subseteq \mathbf{I} - E^{t-1}$  which is not allowed.

Thus if we designate

$$E^t \leftarrow E^{t-1}$$

$$F^t \leftarrow F^{t-1}$$

*Case 3.* There is a (we require that the input requiring that the input these  $|P'_j| \leq t$  values. Equally all values which are in the values in these positions. immediate that  $\forall i \neq j$ ,  $C'_i$

$$E^t \leftarrow E^{t-1}$$

**LEMMA 2.8.** *For any of the input  $\exists$  an integer  $r$*

*Proof.* By induction on  $t = 0$ :  $F^t = \emptyset$  so the claim

Assume the claim for  $t$  (we use additive notation for

$$\begin{aligned} |C^s \cap F^t| &= |C^s \cap (E^t - E^{t-1})| \\ &= |C^s \cap E^t| - |C^s \cap E^{t-1}| \end{aligned}$$

If we only sum over nonempty restricts the input only by  $s + t$  of them. Thus we may

$$\begin{aligned} |C^s \cap F^t| &= |C^s \cap F^{t-1}| \\ &= \frac{p|\mathbf{I}|}{2^{s+t(t-1)/2}} \end{aligned}$$

This follows by the inductive form of the middle terms.

Since all of the denominator is proved.  $\square$

**LEMMA 2.9.**  $e \in E^T$  and

s in the proof of Lemma 2.2. time  $t$ ,  $(S = D^{t-1}, I_e(S) = E^{t-1})$ , be defined by  $h(x) = j$  if  $p_j$  is  $h(x) = 0$  if no one writes on  $x$

processor writes on  $e$ .  $R_j^t$ ). With an almost identical  $R_j^t(y)$ , i.e.  $p_j$  will arrive at the alar,  $p_j$  will write on  $x$  if and

$i \in S_j, z_i = y_i$  if  $i \in S_k - S_j$  and  $E^{t-1}$ . Therefore both  $p_j$  and the CREW PRAM. of the lemma.  $\square$

ne  $x \in I_e(S)$  there exists  $j > 0$  r some  $x \in I_e(S)$ ,  $h(x) = l > 0$ .  $t$ ), and  $S_j^t = S_j - S_l$  for all  $j \in$

$(S)$ , by (5)  $S_j \cap S_l \neq \emptyset$ , and

Since  $x, y \in I_e(S')$ ,  $y \equiv x \equiv h(x) = h(y) = h'(y)$ . an  $i \in S_j^t \cap S_k^t$  s.t.  $x_i \neq y_i$ . Since ever, since  $x \equiv y \equiv e \pmod{S_j}$

$R^t \leq (t-1)t/2$  and  $h'(I_e(S' \cup +1)/2$  and  $h(I_e(S \cup R)) =$

computes a function  $g$  in time  $\leq |I|/r$ . Then  $T = \Omega(\sqrt{\log_2 r})$ . d  $t$ . For any  $j$  let  $P_j^t$  be the set define  $E^t$  and its complement

$E^{t-1}$  such that  $p_j$  writes on  $x$

here is an  $x \in E^{t-1}$  such that which causes a processor  $p_j$  to

t cells since  $|P_j^t| \leq t$ . It is clear  $C_x^t$ . Also any  $y \in E^{t-1} \cap C_x^t$  will not be able to distinguish "cubes" from  $E^{t-1}$  no writes will

The "cubes" also satisfy an additional property. If  $C_x^t \cap C_y^t \neq \emptyset$  then their shared specifying positions must agree in value. Therefore, if  $C_x^t \neq C_y^t$  then  $C_x^t$  and  $C_y^t$  must be specified by different input cells and so correspond to different processors. It follows then that  $C_x^t \cap C_y^t \subseteq I - E^{t-1} = F^{t-1}$  otherwise there would be a simultaneous write which is not allowed.

Thus if we designate the distinct cubes as  $\{C_i^t\}$  then

$$E^t \leftarrow E^{t-1} - \left( \bigcup_i C_i^t \right) \text{ and}$$

$$F^t \leftarrow F^{t-1} \cup \left( \bigcup_i C_i^t \right) \text{ where } \forall i \neq j, C_i^t \cap C_j^t \subseteq F^{t-1}.$$

Case 3. There is a (unique) processor  $p_j$  that writes on  $e \in E^{t-1}$  at time  $t$ : Then we require that the input agree with  $e$  in the positions of  $P_j^t$ . We may regard this as requiring that the input be in the cube which is the subset of the input specified by these  $|P_j^t| \leq t$  values. Equally well this may be regarded as excluding from the input all values which are in the cubes specified by the other  $2^{|P_j^t|} - 1$  possible settings of values in these positions. If we call these excluded cubes  $\{C_i^t\}$  as in Case 2, it is immediate that  $\forall i \neq j, C_i^t \cap C_j^t = \emptyset \subseteq F^{t-1}$ . Then as in Case 2 we have

$$E^t \leftarrow E^{t-1} - \left( \bigcup_i C_i^t \right) \text{ and } F^t \leftarrow F^{t-1} \cup \left( \bigcup_i C_i^t \right).$$

LEMMA 2.8. For any  $t \geq 0$  and any "cube"  $C^s$  which is specified by at most  $s$  cells of the input  $\exists$  an integer  $r$  such that  $|C^s \cap F^t| = r|I|/(2^{s+t(t+1)/2})$ .

Proof. By induction on  $t$

$t = 0: F^t = \emptyset$  so the claim is true with  $r = 0$ .

Assume the claim for  $t-1: F^t = F^{t-1} + \sum_i (C_i^t - F^{t-1})$  since  $\forall i \neq j, C_i^t \cap C_j^t \subseteq F^{t-1}$  (we use additive notation for disjoint union). Therefore

$$\begin{aligned} |C^s \cap F^t| &= \left| C^s \cap (F^{t-1} + \sum_i (C_i^t - F^{t-1})) \right| \\ &= |C^s \cap F^{t-1}| + \sum_i (|C^s \cap C_i^t| - |C^s \cap F^{t-1}|) \\ &= |C^s \cap F^{t-1}| + \sum_i (|C^s \cap C_i^t| - |(C^s \cap C_i^t) \cap F^{t-1}|). \end{aligned}$$

If we only sum over nonempty intersections then  $C^s \cap C_i^t$  is a subset of the input which restricts the input only by specifying input positions and which is specified by at most  $s+t$  of them. Thus we may designate  $C_i^{s+t} = C^s \cap C_i^t$ . Therefore

$$|C^s \cap F^t| = |C^s \cap F^{t-1}| + \sum_i (|C_i^{s+t}| - |C_i^{s+t} \cap F^{t-1}|)$$

$$= \frac{p|I|}{2^{s+t(t-1)/2}} + \sum_i \frac{q_i|I|}{2^{s+t}} - \sum_i \frac{r_i|I|}{2^{s+t+t(t-1)/2}} \text{ where } p, q_i, r_i \text{ are integers.}$$

This follows by the inductive hypothesis for the first and last terms and because of the form of the middle terms.

Since all of the denominators divide  $2^{s+t(t+1)/2}$  the claim holds for  $t$  and the lemma is proved.  $\square$

LEMMA 2.9.  $e \in E^T$  and  $\forall x \in E^T, g(x) = g(e)$ .

*Proof of Theorem 2.3.* If we apply Lemma 2.8 with  $s=0$  then  $C^s = \mathbf{I}$  and we see that  $|F^T|$  is an integral multiple of  $|\mathbf{I}|/2^{T(T+1)/2}$ . Since  $E^T = \mathbf{I} - F^T$  it follows that  $|E^T|$  is also a multiple of this number. Now  $e \in E^T$  so we have  $|E^T| > 0$  and thus  $|E^T| \cong |\mathbf{I}|/2^{T(T+1)/2}$ . By our assumption on  $g$  and by Lemma 2.9 we need  $|E^T| \leq |\mathbf{I}|r$ . Therefore  $2^{T(T+1)/2} \cong r$  and so  $T = \Omega(\sqrt{\log_2 r})$ .  $\square$

**Acknowledgments.** We are indebted to a careful referee who found a serious mistake in our original Lemma 2.3, which claimed a stronger result. We thank Paul Beame for letting us include his results here. We also wish to thank Al Borodin and Dick Lipton for helpful comments on an early version of this manuscript.

## REFERENCES

- [AHU-74] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA 1974.
- [B-83] P. BEAME, personal communication, 1983.
- [CD-82] S. A. COOK AND C. DWORK, *Bounds on the time for parallel RAM's to compute simple functions*, Proc. 14th ACM Symposium on Theory of Computing, 1982, pp. 231-233.
- [CSV-82] A. K. CHANDRA, L. J. STOCKMEYER AND U. VISHKIN, *Complexity theory for unbounded fan-in parallelism*, Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science, 1982, pp. 1-13.
- [FW-78] S. FORTUNE AND J. WYLLIE, *Parallelism in random access machines*, Proc. 10th ACM Symposium on Theory of Computing, 1978, pp. 114-118.
- [FSS-81] M. FURST, J. B. SAXE AND M. SIPSER, *Parity, circuits and the polynomial-time hierarchy*, Proc. 22nd Annual IEEE Symposium on Foundations of Computer Science, 1981, 260-270.
- [GGKMRS-83] A. GOTTLIEB, R. GRISHMAN, C. P. KRUSKAL, K. P. MCAULIFFE, L. RUDOLPH AND M. SNIR, *The NYU ultracomputer—Designing a MIMD Shared Memory Parallel Machine*, IEEE Trans. Comput., C-32 (1983), pp. 175-189.
- [Go-78] L. M. GOLDSCHLAGER, *A unified approach to models of synchronous parallel machines*, Proc. 10th ACM Symposium on Theory of Computing, 1978, pp. 89-94.
- [HU-79] J. HOPCROFT AND J. ULLMAN, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Reading, MA, 1979, pp. 314-318.
- [K-77] D. J. KUCK, *A survey of parallel machine organization and programming*, Computing Surveys, 9 (1977), pp. 29-59.
- [LS-81] R. LIPTON AND R. SEDGWICK, *Lower bounds for VLSI*, Proc. 13th ACM Symposium on Theory of Computing, 1981, pp. 300-307.
- [PS-82] C. PAPADIMITRIOU AND M. SIPSER, *Communication complexity*, Proc. 14th ACM Symposium on Theory of Computing, 1982, 196-200.
- [SV-81] Y. SHILOACH AND U. VISHKIN, *Finding the maximum, merging and sorting in a parallel computation model*, J. Algorithms, 2 (1981), pp. 88-102.
- [ShV-82] ———, *An  $O(\log n)$  parallel connectivity algorithm*, J. Algorithms, 3 (1982), pp. 57-67.
- [SV-82] L. J. STOCKMEYER AND U. VISHKIN, *Simulation of parallel random access machines by circuits*, RC 9362, IBM T. J. Watson Research Center, Yorktown Heights, NY 1982; this Journal, 5 (1984), pp.409-422.
- [V-82] U. VISHKIN, *Parallel-design space distributed—implementation space (PDDI) general purpose computer*, RC 9541, IBM T. J. Watson Research Center, Yorktown Heights, NY, 1982 Theoret. Comput. Sci., to appear.
- [W-83] A. WIGDERSON, *Studies in computational complexity*, Ph.D. thesis, Princeton Univ., Princeton, NJ, May 1983.
- [Y-81] A. YAO, *The entropic limitations on VLSI computations*, Proc. 13th ACM Symposium on Theory of Computing, 1981, pp. 308-311.

## ON THE MOV

JOHN HOP

**Abstract.** The mover's problem is to move one given position to another involving objects with movable in a 3-dimensional region. In this paper a robot arm with an arbitrary moving arm confined within a given time algorithm for moving the arm to a given point within the circle. region of a joint in an arm en

**Key words.** robotics, man

**1. Introduction.** With the problem of designing a robot arm subject to certain geometric constraints, the problem is to determine the path and a constraining region while keeping  $X$  within a region. See [3], [5], [Sch] [Wesley [3], Reif [5], Sch] 2- or 3-dimensional poly

A more difficult problem is to move the object  $X$  has joints and a time algorithm, the degree of freedom for moving  $X$  from position to position with running time polynomial. [5] has shown that the problem is NP-hard. moved from one position to another.

This paper investigates the problem of interest. We begin in §2 with a sequence of objects—ruler—that is, a sequence of objects arises because a natural problem is to move up as compactly as possible. whether an arbitrary carrying object can be folded into a given region. at least NP-hard to decide whether a ruler with one end fixed can be moved within a given 2-dimensional region.

In §§ 4-6 we consider the problem. we are able to give polynomial time algorithms.

\* Received by the editors August 1984, in part by the Office of Naval Research under grant MCS81-01220, and supported by a National Science Foundation Fellowship.

† Computer Science Department

‡ Computer Science Department

§ School of Computer Science