TRADE-OFFS BETWEEN DEPTH AND WIDTH IN PARALLEL COMPUTATION

(PRELIMINARY VERSION)

Uzi Vishkin
Courant Institute
New York University

Avi Wigderson †
Computer Science Division
U C Berkeley

Abstract

A new technique for proving lower bounds for parallel computation is introduced. This technique enables us to obtain, for the first time, non-trivial tight lower bounds for shared-memory models of parallel computation that allow simultaneous read/write access to the same memory location. The size m of the common memory is called **communication width** or **width** in short.

For a wide variety of problems (including parity and majority) 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). This bound is tight fot every $m \leq n/\log^2 n$. We extend our technique to prove $mT^3 = \Omega(n)$ trade-off for a class of "simpler" functions (includind Boolean Or) on a weaker model that forbids simultaneous write access. This result improves the lower bound of Cook and Dwork [CD-82] when communication is limited.

1. Introduction

Consider the following informal problem: there are a large number of people (or processing units), each knows n numbers $a_1, a_2, \cdots, a_n$. 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-Read Exclusive-Write PRAM (CREW PRAM). Both models are precisely defined in section 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 $T=\Omega(\sqrt{n/m})$ 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 non-trivial 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

† This research was conducted while the second author was in the EECS Department at Princeton University.

0272-5428/83/0000/0146$01.00 © 1983 IEEE

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 a new result of Beame [B-83] that achieves a tight bound for the OR function. For a different class of functions (that contains OR), using a different technique, he proves $T=\Omega(\sqrt{n/m})$.

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 section 2. Section 3 contains the upper bounds and section 4 concludes the paper and suggests further research directions. To improve the readability of section 2, the proofs of two technical (but essential) lemmas are given in 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 and 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 $\Pi = \{p_1, p_2, \cdots\}$ of processors, a number n of inputs, n read-only input cells $X(1), X(2), \cdots, X(n)$, *one* common memory cell C, an alphabet Σ and an execution time T.

Each processor p_i has a set of states Q_i and functions

$\rho_i: Q_i \to \{1,2,\cdots,n\}$ - the next input cell to be read,

$\sigma_i: Q_i \to \Sigma$ - the symbol to be written into C, and

$\delta_i: Q_i \times \Sigma \times \Sigma \to Q_i$ - the state transition function.

At each time period $t=0,1,\cdots,T$ 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 $x_i (\in \Sigma)$ $(1 \le i \le n)$, 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

$q_i^{t+1} = \delta_i(q_i^t, X(j), s^t)$ where $j = \rho_i(q_i^t)$, 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 $f(x_1, x_2, \cdots, x_n)$ 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 Π, Σ and Q_i to be infinite, and allow the processors be non-uniform (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 Σ be a set, $I \subseteq \Sigma^n$ and $f : I \to \Sigma$ some function. An input $x = x_1, x_2, \cdots, x_n \in I$ is said to be *k-sensitive* w.r.t f if for every subset $J \subseteq \{1,2,\cdots,n\}$, $|J| = k-1$ there exists another input $y = y_1, y_2, \cdots, y_n \in I$ 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) $x \in I$ 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, OR : $\{0,1\}^n \to \{0,1\}$.
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 the 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. ▪.

Let us informally discuss the difficulties we are facing in trying to prove theorem 2.1. Consider the behaviour 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_j) 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 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 Σ 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 (non empty) set of all possible inputs (the domain). Fix a time period t and let $\beta = s^0 s^1 \cdots s^{t-1}$ be the string of succesive symbols in C in time periods $0,1,\ldots,t-1$. β is called the history through time t. Denote by $\mathbf{I}_\beta \subseteq \mathbf{I}$ the subset of inputs that have history β through time t.

Our analysis will be based on the observation that cases 1 and 2 consist each of two subcases. Fix β, 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 β itself contains the information that no one will write at time t. Similarly, in case 2a, β 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 $\mathbf{I}(\{(i_1,y_1),\cdots,(i_l,y_l)\}) = \{ x \in \mathbf{I} \mid x_{i_j} = y_j,\ 1 \le j \le l \}$ denote the set of all inputs (n-tuples) whose projection on the l-tuple $(i_1,i_2,\cdots,i_l)$ is $(y_1,y_2,\cdots,y_l)$.

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 $D^0 = \phi$. 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

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

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 $i_1,i_2,\cdots,i_u$ and $y_{i_1},y_{i_2},\cdots y_{i_u}$ be the sets of input cells and their contents (respectively) that were read by p_l up to time t. (Clearly $u \le t$). Then

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

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

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

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

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

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

In particular we have:

Lemma 2.1: $E^T \neq \phi$ 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 $T=\Omega(\sqrt{k})$. ▪

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 ith 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. $I=\Sigma^n$). The main result in this subsection is:

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 $T=\Omega(n^{1/3})$

Recently we received a communication from Beame [B-83], that improves the result of corollary 2.2.

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

The proof of this theorem will not be given here. An immediate corollary of it is:

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

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 $g : I \to \Sigma$ be the function being computed and $e=e_1,e_2,\cdots,e_n \in I$ be a k-sensitive input w.r.t. g. Set $E^0=I$ and $D^0=\phi$. 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

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

Case 2: There is a (unique) processor p_j that writes on $e \in E^{t-1}$ at time t. Let $i_1,i_2,\cdots,i_u$ and $e_{i_1},e_{i_2},\cdots e_{i_u}$ 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

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

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

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:

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

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

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. ▪

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. ▪

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. ▪

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 $C(1), C(2), \cdots, C(m)$ 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 2.3:
Let M be a CRCW PRAM(m) that computes a k-sensitive everywhere function $f:I(\subseteq\Sigma^n)\rightarrow\Sigma$ in time T. Then $T=\Omega(\sqrt{k/m})$.
In particular, if $f\in${Parity, Majority, Sum, Max}, $T=\Omega(\sqrt{n/m})$.
Let N be a CREW PRAM(m) that computes a k-sensitive somewhere function $g:\Sigma^n\rightarrow\Sigma$. Then $T=\Omega((k/m)^{1/3})$.
In particular, if $g\in${AND, OR}, $T=\Omega((n/m)^{1/3})$

The theorem of Beame that was mentioned in the previous subsection can also be extended to arbitrary communication width, and for the OR function improves our lower bound:

Theorem (Beame): If the OR function on n bits is computed by a CREW PRAM(m) in time T, then $T=\Omega(\sqrt{n/m})$.

The only difficulty in extending our technique to prove theorem 2.3 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 ith slice only cell $C(i)$ may be written into. Then, at the ith 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 cell 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_i} and D^{t_i}, $0\leq t\leq T$, $1\leq i\leq m$ inductively in a similar fasion 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_i} from $E^{t_{i-1}}$ when $i>1$, and E^{t_1} from $E^{(t-1)_m}$.

The analysis of the previous subsections carries through in a straightforward manner w.r.t the final sets, E^{T_m} and D^{T_m}. 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), $|D^{T_m}|\leq m(T+1)T/2$.
In the CREW PRAM(m), $|D^{T_m}|\leq mT(T+1)(T+2)/6$.

Lemma 2.7:
In the CRCW PRAM(m), for every $x,y\in E^{T_m}$, $f(x)=f(y)$.
In the CREW PRAM(m), for every $x,y\in E^{T_m}$, $g(x)=g(y)$.

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 $a_1, a_2, \cdots, a_n$ are initially stored in the read-only input tape. Let L_j be a local memory cell of processor p_j, and C is the common memory cell. The algorithm for $n=10$ is described in Figure 1.

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

Figure 1: Summation with one common memory cell

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 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) The lower bounds for the OR on the CREW PRAM given here, combined with that of Cook and Dwork, covers the whole range of m. On the other hand, the lower bound for the parity function 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 parallel RAMs in which processors are allowed to be probabilistic or non-deterministic. In the deterministic version of the CRCW PRAM(1) which we studied here, both the Parity and the Max functions have an $\Omega(\sqrt{n})$ lower bound on the time. If we allow non-determinism, the maximum of n numbers can be computed in constant time. However, we conjecture that the lower bound still holds for Parity even in the non-deterministic model.

(3) Study Time - Width - Processors trade-offs for other functions.

Acknowledgements

We are indebted to a careful referree 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 for helpful comments on an early version of this manuscript.

Appendix

Lemma 2.2: For every $x,y\in E^T$, $f(x)=f(y)$.

Proof of lemma 2.2: We use the following notation. For an input $x\in I$,

- $q_i^t(x)$ and $s^t(x)$ are respectively the state of p_i and the contents of C in time t for the input x.
- $R_j^t(x) = \{\rho_j(q_j^r(x)) \mid 0\leq r<t\}$ is the set of input cells read by p_j through time t. Set $R_0^t(x)=\phi$.
- $w^t(x)$ is the index of the processor that writes at time t on input x. If there is no such processor at time t for x, $w^t(x)=0$.
- $W^t(x) = \bigcup_{r=1}^{t} R_j^r(x)$ where $j=w^r(x)$, is the set of input cells read by all writers through time t.
- $FW^t(x) = \{ w^r(x) \mid t\leq r\leq T, w^r(x)\neq 0 \}$ is the set of future writers from time period t on.

Let x and y be elements in E^T. It is sufficient to show that $s^T(x)=s^T(y)$. We prove by induction on t, that $w^t(x)=w^t(y)$, $W^t(x)=W^t(y)$, $s^t(x)=s^t(y)$, and that for every $j\in FW^t(x)$ $q_j^t(x)=q_j^t(y)$ and $R_j^t(x)=R_j^t(y)$.

$t=0$. For every processor j, $q_j^0(x)=q_j^0(y)=q_j^0$, $R_j^0(x)=R_j^0(y)=\phi$. Also, $s^0(x)=s^0(y)=b_0$, $W^0(x)=W^0(y)=\phi$ and $w^0(x)=w^0(y)=0$.

$t>0$. Assume the claim holds for every $r<t$. Let $j\in FW^t(x)$ Let $i(x)=\rho_j(q_j^{t-1}(x))$, $i(y)=\rho_j(q_j^{t-1}(y))$. By

the induction hypothesis $i(x)=i(y)$ and hence $R_j^t(x)=R_j^t(y)$. Since $j\in FW^t(x)$, $R_j^t(x)\subseteq W^T(x)$ which using $x,y\in E^T$ implies that $x_{i(x)}=y_{i(x)}$. From this and the induction hypothesis we get $q_j^t(x)=q_j^t(y)$. Let $\sigma_j(x)=\sigma_j(q_j^t(x))$ and $\sigma_j(y)=\sigma_j(q_j^t(y))$. Then $\sigma_j(x)=\sigma_j(y)$. There are two cases to consider now,

case 1: $s^t(x)=s^{t-1}(x)$. By the construction of E^t and induction, $s^t(y)=s^t(x)=s^{t-1}(x)$, $w^t(x)=w^t(y)=0$ and $W^t(x)=W^t(y)=W^{t-1}(x)$.

case 2: $s^t(x)\neq s^{t-1}(x)$. Let $j=w^t(x)$. Again by construction of E^t, there can be no $l<j$ s.t. $\sigma_l(q_l^t(y))\neq s^{t-1}(y)$, and since $\sigma_j(x)=\sigma_j(y)$ we have $w^t(y)=j=w^t(x)$, $s^t(y)=s^t(x)$, and $W^t(y)=W^t(x)$. ▪

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

Proof of lemma 2.3: Denote by Z^+ the set of nonnegative integers, and i,j,k,l denote only positive integers. Also, for a subset $S\subseteq\{1,2,\cdots,n\}$ and inputs $x,y\in\mathbf{I}$,

- $x\equiv y(mod\ S)$ means $x_i=y_i$ for all $i\in S$).
- $\mathbf{I}_y(S)=\{x\in\mathbf{I}\mid x\equiv y(mod\ S)\}$.

Claim: Given an integer t, a set $S\subseteq\{1,2,\cdots,n\}$, a function $h:\mathbf{I}_e(S)\to Z^+$, and sets $S_j\subseteq\{1,2,\cdots,n\}$ for every positive $j\in h(\mathbf{I}_e(S))$ that satisfy

(1) $S_j\cap S=\phi$ for all j.

(2) $|S_j|\leq t$ for all j.

(3) $h(e)=0$.

(4) $h(x)=j$ and $y\equiv x(mod\ S_j)$ implies $h(y)=j$.

(5) $h(x)=j$, $h(y)=k$ and $j\neq k$ implies that there exists an $i\in S_j\cap S_k$ s.t. $x_i\neq y_i$.

Then there exists a set $R\subseteq\{1,2,\cdots,n\}$ s.t. $|R|\leq t(t+1)/2$ and $h(\mathbf{I}_e(S\cup R))=0$.

Connection between the claim and the lemma. Recall that we wanted to prove the existance of $t(t+1)/2$ input positions s.t. fixing them with values of e will ensure that no one writes at time t in case 3. Let R_j^t be defined as in the proof of lemma 2.2. Then let S be the set of fixed input positions through time t, $(S=D^{t-1}, \mathbf{I}_e(S)=E^{t-1})$, $S_j=R_j^t-S$ for all j, and let the function $h:\mathbf{I}_e(S)\to Z^+$ 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) $|s^t|=|R_j^t-S|\leq|R_j^t|\leq t$

(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 $x\equiv y(mod\ S_j)$, $x\equiv y(mod\ R_j^t)$. With an almost identical proof to that of lemma 2.2 we can prove that $q_j^t(x)=q_j^t(y)$, 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, contradiction to the definition of the CREW PRAM.

Now we can take $R_0^t=R$, which completes the proof of the lemma. ▪

Proof of Claim: The proof is by induction on t.

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

$t>0$. If $h(\mathbf{I}_e(S))\equiv\{0\}$ we are done. Assume that for some $x\in\mathbf{I}_e(S)$, $h(x)=l>0$. Set $S'=S\cup S_l$, let 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') $S'\cap S_j'=\phi$ for all j. Clear.

(2') $|S_j'|\leq t-1$ for all $j\in h'(\mathbf{I}_e(S'))$. Since $l,j\in\mathbf{I}_e(S))$, by (5) $S_j\cap S_l\neq\phi$, and therefore $|S_j'|=|S_j-S_l|\leq$ $\leq|S_j|-1\leq t-1$.

(3') $e\in\mathbf{I}_e(S')$ and $h'(e)=0$. Clear.

(4') $h'(x)=j$, $y\equiv x(mod\ S_j')$ implies $h'(y)=j$. Since $x,y\in\mathbf{I}_e(S')$, $y\equiv x\equiv e(mod\ S_l)$ and therefore $y\equiv x(mod\ S_j)$. 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 $x\equiv y\equiv e(mod\ S_l)$, 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 $h(\mathbf{I}_e(S\cup R))=$ $=h(\mathbf{I}_e(S'\cup R'))=h'(\mathbf{I}_e(S'\cup R'))=0$. ▪

References

[AHU-74] Aho, A.V., Hopcroft, J.E. and Ullman, J.D., *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA 1974.

[B-83] Beame, P., Private Communication.

[CD-82] Cook, S.A. and Dwork, C. "Bounds on the time for parallel RAM's to compute simple functions", *Proc. 14th ACM Symposium on Theory of Computing*, 1982, 231-233.

[CSV-82] Chandra, A.K., Stockmeyer, L.J. and Vishkin, U., "Complexity Theory for unbounded fan-in parallelism", *Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science*, 1982, 1-13.

[FW-78] Fortune, S. and Wyllie, J., "Parallelism in Random Access Machines", *Proc. 10th Symposium on Theory of Computing*, 1978, 114-118.

[FSS-81] Furst, M., Saxe, J.B., and Sipser, M., "Parity, circuits and the polynomial-time hierarchy", *Proc. 22nd Annual IEEE Symposium on Foundations of Computer Science*, 1981, 260-270.

[GGKMRS-83] Gottlieb, A., Grishman, R., Kruskal, C.P., McAuliffe K.P., Rudolph, L. and Snir, M., "The NYU Ultracomputer - Designing a MIMD Shared Memory Parallel Machine", *IEEE Trans. on Computers*, C-32, 2, 1983, 175-189.

[Go-78] Goldschlager, L.M., "A Unified Approach to Models of Synchronous Parallel Machines," *Proc. 10th ACM Symposium on Theory of Computing*, 1978, pp. 89-94.

[HU-79] Hopcroft, J. and Ullman, J., "*Introduction to Automata Theory, Languages and Computation*", Addison-Wesley, 1979, 314-318.

[K-77] Kuck, D.J., "A survey of parallel machine organization and programming", *Computing Surveys*, 9, 1 (1977), pp. 29-59.

[LS-81] Lipton, R. and Sedgwick, R. "Lower Bounds for VLSI", *Proc. 13th Symposium on Theory of Computation*, 1981, 300-307.

[PS-82] Papadimitriu, C. and Sipser, M., "Communication Complexity", *Proc. 14th Symposium on Theory of Computing*, 1982, 196-200.

[SV-81] Shiloach, Y., and Vishkin, U., "Finding the maximum, merging and sorting in a parallel computation model", *J. of Algorithms*, 2,1 (March 1981), pp. 88-102.

[ShV-82] Shiloach, Y. and Vishkin, U. "An $O(\log n)$ Parallel Connectivity Algorithm", *J. of Algorithms*, 3,1 (1982), pp 57-67.

[SV-82] Stockmeyer, L.J. and Vishkin, U., "Simulation of parallel random access machines by circuits", RC 9362, IBM T.J. Watson Research Center, Yorktown Heights, N.Y. 1982. To appear in *SIAM J. on Computing*

[V-82] Vishkin, U., "Parallel-Design Space Distributed - Implementation Space (PDDI) General Purpose Computer", RC 9541, IBM T.J. Watson Research Center, Yorktown Heights, N.Y. 10598, 1982.

[Y-81] Yao, A., "The Enthropic Limitations on VLSI comutations", *Proc. 13th Symposium on Thory of Computing*, 1981, 308-311.