Limits on the Power of Concurrent-Write Parallel Machines

Paul Beame

Dept. of Computer Science
University of Toronto
Toronto, Canada
M5S 1A4

Abstract

We prove lower bounds for the computation of simple functions on generalized versions of parallel random access machines which allow both concurrent reads and concurrent writes. In particular we show that if the number of processors is limited by a polynomial in n then computing the sum of n n-bit integers requires time $\Omega(\log n)$ and computing the parity of n input bits requires time $\Omega(\sqrt{\log n})$. The latter result, using reductions given by Chandra, Stockmeyer, and Vishkin (1984), implies that a host of problems including sorting or adding n input bits, or multiplying two $n/2$-bit integers also require time $\Omega(\sqrt{\log n})$ to compute.

1. Introduction

Parallel random access machines (PRAM's) are well-accepted as good models for parallel computation. The procedural nature of the way in which they compute and the relatively natural way of enforcing uniformity conditions on them have made them popular for the design of parallel algorithms. They consist of many processors acting in consort and communicating through some shared memory. They operate much like familiar sequential RAM's except for the rules for concurrently accessing the shared memory. There are three main classifications of these rules: exclusive read - exclusive write (EREW), concurrent read - exclusive write (CREW), and concurrent read - concurrent write (CRCW). The most powerful of these machines (CRCW) can be simulated by the weakest (EREW) with a delay per step proportional to the logarithm of the number of processors.

Cook, Dwork and Reischuk (1984) have shown that either the EREW or CREW PRAM's require $\Theta(\log n)$ time to compute the OR of n bits. Their lower bound holds independently not only of the number of processors or the size of the shared memory but even of the way the program is specified and the instruction set of a processor. It really says something about the restrictive nature of the communication itself.

The OR of n bits can easily be computed in constant time on a CRCW PRAM and it is easy to see how with sufficiently many processors and cells any boolean function can be computed in constant time. It is natural then to consider CRCW PRAM's which have resources bounded by a polynomial in the size of the input and try to prove lower bounds similar to those for PRAM's which only have exclusive writes. Previous lower bounds for CRCW PRAM's have approached this in different ways. Some have put extremely stringent restrictions on the number of processors or the size of the shared memory, e.g. Vishkin and Wigderson (1985), or Fich, Meyer auf der Heide, Ragde and Wigderson (1985). Other bounds which hold with a polynomial number of processors or cells but which put severe restrictions on the instruction sets of the PRAM's are due to Stockmeyer and Vishkin (1984) and Meyer auf der Heide and Reischuk (1984). Other more powerful primitives than those they allow seem to be perfectly reasonable since the cost of concurrent accesses is presumably significantly greater than that of local computation. A symptom of the restricted nature of these CRCW PRAM's is that most Boolean functions

require time $\Omega(\frac{n}{\log n})$ to compute given a polynomial number of processors and cells whereas all Boolean functions can be computed in $O(\log n)$ time by the EREW or CREW machines of Cook, Dwork and Reischuk using only n processors and cells.

We consider computation of specific functions for the most general form of CRCW machine which we call the Abstract CRCW PRAM. We first show that such a machine can compute any function of Boolean inputs in time $\log n - \text{loglog } n + O(1)$ given a polynomial number of processors and cells and that this bound is tight. Essentially the same machines are considered by Meyer auf der Heide and Wigderson (1985) in which the authors prove an $\Omega(\sqrt{\log n})$ time lower bound for sorting n integers and a $\Theta(\log n)$ lower bound for adding n integers. Their results rely on Ramsey theory and as a consequence their bounds only hold for integers which are extremely large - so large that the problems in question are not polynomial time computable given a sequential machine with an honest complexity measure like the log-cost RAM.

We show that on the Abstract CRCW PRAM the sum of n numbers requires $\Theta(\log n)$ time even if for example the numbers have only n bits each. Using a result of Hastad (1986) concerning unbounded fan-in circuits, the main bound we prove is that on this extremely powerful model some very simple functions on $\{0,1\}^n$ require time $\Omega(\sqrt{\log n})$ to compute given a polynomial bound on the number of cells and processors. These functions include computing the parity of, sorting, or adding n bits as well as multiplying two $n/2$-bit integers. These results are the first non-trivial lower bounds for computing Boolean functions on CRCW PRAM's which do not rely on restricted instruction sets of processors or resources smaller than the size of the problem input. They show that there is something inherent in the ways in which processors communicate with each other which limits their computational power.

2. The Abstract CRCW PRAM

Definition:

An *Abstract CRCW PRAM* is a shared memory machine with processors $P_1, \ldots, P_{p(n)}$ which communicate through memory cells $X_1, \ldots, X_{c(n)}$. The input is initially stored in the first n cells of memory, $X_1, \ldots, X_n$. Initially all cells other than the input cells contain the value 0. The output of the machine is the value in the cell X_1 at time $T(n)$.

Before each step t processor P_i is in state q_i^t. At time step t, depending on q_i^t, processor P_i reads some cell X_j of shared memory, then, depending on the contents (X_j) and q_i^t, assumes a new state q_i^{t+1} and depending on this state, writes a value $v = v(q_i^{t+1})$ into some cell.

When several processors are attempting to write into a single cell at the same time step the one that succeeds will be the lowest numbered processor.

A processor may write anything into its cells when it writes, including a full description of the portion of the history of the computation which it knows, as well as its own processor number.

In the definition of an Abstract CRCW PRAM it becomes clear that, except for the final step of an algorithm in which the machine must output function values, the actual values of a processor's state or the contents of a cell are not important. What is important is the set of inputs which lead to the cell contents or state. The computation then may be viewed as operating not on actual values so much as on the partitions associated with them.

Definition: Let M be an Abstract CRCW PRAM. For any processor P_i the *processor partition*, $P(M, i, t)$, of the input set at time step t is defined so that two inputs are in the same equivalence class of $P(M, i, t)$ if and only if they lead to the same state of processor P_i at the end of time step t.

For any cell X_j the *cell partition*, $C(M, j, t)$, of the input set at time t is defined so that two inputs are in the same equivalence class of $C(M, j, t)$ if and only if they lead to the same contents of cell X_j at the end of time step t.

The idea that these processor and cell partitions are the crucial aspects of a computation is implicit in much of the lower bound work in this area. For example, Snir (1985) has given a formal explanation of what a most powerful EREW PRAM can do which is essentially a method of describing the ways that the processor and cell partitions of that machine may be combined during a computation.

It will be useful to use the following restriction of the Abstract CRCW PRAM in order to simplify our discussions of these machines.

Definition: We say that an Abstract CRCW PRAM *satisfies the common write rule* if whenever several processors are attempting to write into a single cell at the same time step, the values that they are attempting to write are the same.

Lemma 2.1: [Kucera (1982)] Let M be a general Abstract CRCW PRAM with $p(n)$ processors, $c(n)$ memory cells and taking $T(n)$ time. Then an Abstract CRCW PRAM which satisfies the common write rule can simulate M by using $O(p(n)^2)$ processors, $c(n) + p(n)$ memory cells and taking $4T(n)$ time.

3. The Computation of Arbitrary Boolean Functions and Integer Addition

From the Lupanov bounds on combinational circuit complexity (see Savage (1976)), the obvious simulation of unbounded fan-in circuits by combinational circuits, and the simulations in Stockmeyer and Vishkin (1984) we see that restricted CRCW PRAM's require time $\Omega(\frac{n}{\log n})$ to compute most Boolean functions given a polynomial number of processors. We now see that an Abstract CRCW PRAM has much greater computational power than CRCW PRAM's with restricted instruction sets.

Theorem 3.1: An Abstract CRCW PRAM which satisfies the common write rule and has $c(n) \geq p(n) \geq n$ can compute any function of Boolean inputs in time $\log n - \text{loglog}(\frac{p(n)}{n}) + O(1)$.

Proof: Suppose there are $p(n) = n\,2^k$ processors. We exhibit an algorithm which computes the function in $\log n - \log k + 3$ steps.
Break up the input into chunks of k bits each. Assign $k\,2^k$ processors to each chunk, and associate a label (i,α) with each processor for each $\alpha \in \{0,1\}^k$ and each $i = 1, \ldots, k$.
(1) In the first step each processor with label (i,α) reads the i-th bit within its chunk and writes a 1 into a cell labelled α for its chunk if and only if the value it read disagreed with the i-th bit of α.
(2) In the second step one processor for each α within each chunk reads cell α and if it reads a 0 it writes α into a single cell designated for that chunk.
The input has now been effectively compressed from n bits in n cells to $\frac{n}{k}$ k-bit integers in $\frac{n}{k}$ cells. The remainder of the algorithm now uses a standard binary fan-in of the $\frac{n}{k}$ cells which contain the description of the input to arrive at the situation where a complete description of the input is contained in one cell.
This takes $\left\lceil \log \frac{n}{k} \right\rceil$ steps since there is already one processor which knows the contents of each cell.
Processor P_1 now reads this cell and writes the value of the function into X_1.
Since $\log \frac{n}{k} = \log n - \log k$ the running time is as claimed. □

Let $b \geq 2$ and $\Pi = \{0,1, \ldots, b-1\}$. Let f_b be the function on Π^n which converts an input string into the integer which it represents in base b. For inputs from Π^n this is the hardest function to compute on the Abstract CRCW PRAM since its output provides a complete description of the input set and in one further step any processor can read the first cell and compute any other function.

Theorem 3.2: An Abstract CRCW PRAM to compute the function f_b described above requires $T(n) \geq \log n + 1 - \log[\,1 + \log_b 2p(n)\,]$.

Proof: The important fact about the definition of f is that when it has been computed the contents of the first cell must induce a partition of the inputs which has b^n distinct classes.

Let P_t (C_t) be a least upper bound over all processors (cells) of the number of classes in the processor (cell) partitions induced on the input set at the end of time step t. Since the input is initially present in the first n cells and since the processors initially have no access to it, it is clear that:

$P_0 = 1$ and $C_0 = b$

The cell which a processor reads during time step $t+1$ can only depend on its state at time t so that on elements of one class in the partition only one cell may be read. Thus each class in the partition at time t may be split into at most C_t classes during time step $t+1$ since this is the maximum number of classes distinguishable for a cell at time t. It follows that:

$P_{t+1} = P_t\, C_t$

A cell may have all $p(n)$ processors writing into it during step $t+1$ and each processor may communicate the entire partition of the portion of the input on which it succeeds in writing into the cell. Also the cell may still maintain the partition of the portion of the input on which no processor writes. Thus the number of classes into which the contents of the cell may resolve the input satisfies:

$C_{t+1} = p(n)\, P_{t+1} + C_t$

Easy calculation shows that for $t \geq 1$ we can bound C_t by $(2p(n)b)^{2^{t-1}}$.

In order to compute f_b in T steps, $C_T \geq b^n$ is needed. Therefore $2^{T-1}[1+\log_b 2p(n)] \geq n$ and so $T \geq \log n + 1 - \log[1+\log_b 2p(n)]$. □

By choosing $b=2$ in Theorem 3.2 we see that the bound in Theorem 3.1 is tight. Using Theorem 3.2 it is an easy step to prove a lower bound for the addition of integers.

Corollary 3.1: The sum of n integers of $n \log b$ bits each requires time at least $\log n + 1 - \log[1+\log_b 2p(n)]$ to compute on an Abstract CRCW PRAM with $p(n)$ processors.

Proof: The function f_b is exactly the function computed by considering the i-th input x_i as $x_i\, b^{i-1}$ and adding the resulting integers. The Corollary follows immediately since the largest such integer is bounded by b^{n-1} and so requires only $n \log b$ bits. □

Corollary 3.2: The sum of n integers with $\omega(n \log n)$ bits each requires time $\log n$ to compute on an Abstract CRCW PRAM given a number of processors polynomial in n.

Corollary 3.3: The sum of n integers with n bits each requires time $\Omega(\log n)$ to compute on an Abstract CRCW PRAM with as many as 2^{n^ϵ} processors for any $\epsilon<1$.

A $\log n$ lower bound for integer addition on similar machines has also been proved by Parberry (1985) but the integers must have more than polynomially many bits in n for this to hold for all machines with a polynomial number of processors and cells.

It is interesting to note that, since the sum of n integers with polynomially many bits has $O(\log n)$ depth using combinational circuits, each bit of the output of the sum can be computed in time $O(\frac{\log n}{\log\log n})$ on a CRCW PRAM. It is merely the requirement that the entire output must appear in one cell which is responsible for the additional complexity.

4. Partitions, Lengths and Projections

We may describe each equivalence class in a partition of the input set $I = \{0,1\}^n$ by a Boolean formula which expresses the characteristic function of that equivalence class. When this formula is written in disjunctive normal form (DNF), the subset of the inputs which it describes may be written as the union of inputs which satisfy each clause.

Definition: For any partition A of $J \subseteq I = \{0,1\}^n$ let the *length of A*, $l(A)$ be the length of the longest clause in a minimal DNF formula for the characteristic function of each equivalence class in A when considered as a subset of J.

Remark: If a partition B is a refinement of partition A then $l(A) \leq l(B)$.

A *projection* π of the input set I is a map:

$$\pi : \{1,2,\ldots,n\} \rightarrow \{0,1,*\}$$

where

$$\pi(i) = \begin{cases} 1 \text{ means } x_i \text{ is set to 1} \\ 0 \text{ means } x_i \text{ is set to 0} \\ * \text{ means } x_i \text{ is unset} \end{cases}$$

Definition: For any partition A of I and any projection π let A^π be the restriction of A to the set $I^\pi = \{x \in I : \forall x_i \text{ set by } \pi,\ x_i = \pi(i)\}$.
If F is a Boolean formula then F^π is the formula obtained by replacing each literal corresponding to an input which π sets by the truth value assigned by $\pi(i)$.

Lemma 4.1 Let π be a projection and A a partition of I. If F is a DNF formula for the characteristic function of some equivalence class $C \in A$ then F^π is a DNF formula for the characteristic function of the corresponding equivalence class $C^\pi \in A^\pi$ considered as a subset of I^π.

Definition: A *random restriction* ρ chosen from R_q is a function which independently assigns 0, 1, or * to each $i \in \{1,2, \ldots, n\}$, where:

$$\Pr[\ \rho(i) = *\ ] = q$$

$$\Pr[\ \rho(i) = 1\ ] = \frac{1}{2}(1-q)$$

$$\Pr[\ \rho(i) = 0\ ] = \frac{1}{2}(1-q)$$

Let π be a projection of the input set I. A *random q-specialization* π' of π is a projection which agrees with π on all the inputs set by π and which takes the values set by a randomly chosen $\rho \in R_q$ on the remaining inputs.

An important measure of the progress in the computation of an Abstract CRCW PRAM will be $l(A^{\pi})$ where A is $P(M, i, t)$ or $C(M, j, t)$ and π is an appropriate projection.

5. Lower Bounds for Some Simple Functions

The parity function is the function on binary values $x_1, \ldots, x_n$ which produces their sum modulo 2. Furst, Saxe and Sipser (1984) gave a $\Omega(\log^* n)$ lower bound on the depth of polynomial size unbounded fan-in circuits computing parity. Ajtai, extending the results in Ajtai (1983), and Babai (1984), improved the depth lower bound for polynomial size parity circuits to $\Omega(\sqrt{\log n})$. Independently, by applying and modifying the techniques of Furst, Saxe and Sipser (1984), we were able to derive an intermediate lower bound which also applies to the Abstract CRCW PRAM model described here.

Theorem 5.1: [Beame (1985)] If M is an Abstract CRCW PRAM computing the parity function with $c(n) = n^{O(1)}$ and $p(n)$ unbounded then $T(n) = \Omega(\sqrt{\log\log n})$.

The results for unbounded fan-in circuits were based on efforts to achieve exponential lower bounds for constant depth circuits computing parity. Yao, in a breakthrough paper (Yao (1985)), was able to give exponential lower bounds for constant depth parity circuits but his results do not seem to imply anything better than $\Omega(\sqrt{\log n})$ lower bounds for polynomial size parity circuits. Also, because of the notion of approximation, Yao's proofs do not appear to translate well to the machine model with which we are concerned.

Using some of the techniques in Yao (1985), Hastad (1986) has obtained improved lower bounds for parity circuits. His improved results yield $\Omega(\frac{\log n}{\log\log n})$ lower bounds for polynomial size parity circuits which match the upper bounds for such circuits given by Chandra, Stockmeyer and Vishkin (1984). He also eliminates the necessity for approximation as used by Yao. This makes his proofs amenable to modification and application to obtain the following result which has a much shorter proof than Theorem 5.1.

Theorem 5.2: If M is an Abstract CRCW PRAM which computes the parity function with $p(n) = n^{O(1)}$ and $c(n)$ unbounded then $T(n) = \Omega(\sqrt{\log n})$.

Proof: We assume without loss of generality by Lemma 2.1 that M satisfies the common write rule since the simulation only squares the number of processors. We will also assume that $p(n) \geq n$ and when processors write a value they tag it with the time step during which they are writing. This does not conflict with the common write and can only transmit additional information.

We will define a projection π_t for each step t of the computation such that after step t and after π_t is applied, the cell partitions will all have length less than the number of unset bits. The lower bound will follow since parity has minimal DNF clauses which depend on *all* the unset bits.

Define $\pi_0, \pi_1, \ldots$ as follows:

$\pi_0(i) = *$ for all i (all bits are unset).

π_t will be chosen from the random q_t-specializations of π_{t-1}, where q_t will be defined later.

Let $p_t = \max_i l(P(M, i, t)^{\pi_{t-1}})$

[when $t = 0$, consider $\pi_{t-1} = \pi_0$],

$c_t = \max_j l(C(M, j, t)^{\pi_t})$,

and $m_t =$ the number of bits unset by π_t.

Let $b_t = 3^t \log_n p(n)$

and $q_t = \frac{1}{20\, b_t} p(n)^{-1/b_t} = \frac{1}{20\, b_t} n^{-3^{-t}}$.

Claim: For $t \geq 0$ we can choose π_t so that $p_t \leq b_t$, $c_t \leq 2b_t$ and $m_t \geq n\ 2^{-t} \prod_{i=1}^{t} q_i$.

We show this by induction:
Base case: $p_0 = 0 < c_0 = 1 \leq b_0$ and $m_0 = n$.

Induction step: Let $t \geq 1$. Since the cell which a processor reads during step t is dependent solely upon its current state, the new state which the processor assumes will depend only on the old state and the equivalence class in which the value in the cell read lies. Since the clauses in the cell partition have only c_{t-1} literals corresponding to unset input bits, the longest DNF clause describing the new state's equivalence class will have at most $p_{t-1} + c_{t-1}$ literals corresponding to input bits which were unset after step t. Thus it follows that:

$$p_t \leq p_{t-1} + c_{t-1} \leq 3b_{t-1} = 3^t \log_n c(n) = b_t .$$

From this recurrence it is easy to see that the concurrent reads do not give the Abstract CRCW PRAM much of its power. An identical recurrence would also hold for CREW PRAM's. It is the concurrent writes which cause all the difficulty.

We now try to bound c_t.
Cells into which no processor writes during step t on any input in $I^{\pi_{t-1}}$ will be unaffected by the write step and so will have equivalence classes with clauses bounded by c_{t-1}. Thus we only need to consider cells into which processors may write on some input in $I^{\pi_{t-1}}$. We say that such cells are used during write step t.

For the cells used during write step t, we first consider equivalence classes which correspond to values written by processors during step t. Because M satisfies the common write rule, the set of inputs on which some value v is written into a particular cell is the union of the sets of inputs on which each processor writes v into that cell. In fact, because of the tagging by time step during writes, the equivalence class in the cell corresponding to v is exactly such a union. Since the set of inputs on which a particular processor performs any action is a union of equivalence classes in that processor's partition, the set of inputs on which some processor writes v into a particular cell is a union of equivalence classes of processors. Thus the equivalence class corresponding to v is a union of classes with DNF clauses bounded by p_t, and so its DNF clauses have maximum length at most $p_t \leq b_t$.

For each cell used during write step t it remains to consider the equivalence classes which correspond to the cases when no processor has actually written into the cell during step t. Each such class is the intersection of some old cell class and the set of inputs on which no processor writes into the cell during step t. We will choose π_t in such a way that the set of inputs on which no processor writes into the cell is described by a DNF formula whose clauses are bounded by b_t. Then each class within the cell will have DNF clauses bounded by $c_{t-1} + b_t$. To achieve this we notice that the set of inputs on which some processor *writes* into a cell is a union of processor classes and so is already described by a DNF formula with clauses bounded by p_t. Then it follows that G, the formula describing the set of inputs on which *no* processor writes into the cell, is the negation of a formula with short DNF clauses and so has CNF clauses bounded by $p_t \leq b_t$. We now apply the following lemma which is essentially the main lemma of Hastad (1986).

Lemma 5.1: [Hastad (1986)] Let G be any CNF formula each of whose clauses has at most r literals. Let ρ be a random restriction chosen from R_q. Then for appropriate $\alpha < 5qr$,

$$\Pr[\ G^{\rho} \text{ has minimal DNF with any clause size} \geq s\ ] \leq \alpha^s$$

Thus if we choose a random q_t-specialization ρ of π_{t-1}

$$\Pr[\ l(\ C(M, j, t)^{\rho}\ ) \geq c_{t-1} + b_t\ ] < (5\ q_t\ b_t)^{b_t}$$

$$= \left(\frac{1}{4} p(n)^{-1/b_t}\right)^{b_t}$$

$$= 4^{-b_t} p(n)^{-1}.$$

It is clearly only necessary to apply Hastad's lemma once per cell used during write step t. An upper bound on the number of cells used during write step t is certainly the number of possible states of all processors at the time of the write. This is just the number of classes in the processor partitions at the time of the write. Because the DNF clauses describing each equivalence class are bounded by $p_t \leq b_t$ and a class is

the union of the set of inputs satisfying each DNF clause, each class contains a fraction of at least 2^{-b_t} of the possible inputs. The classes within a single processor partition are disjoint so there at most 2^{b_t} classes per processor. Thus at most $p(n)2^{b_t}$ cells are used during write step t. It follows that

$$\Pr[\text{ for any cell } X_j,\ l(\ C(M, j, t)^{\rho}\ ) \geq c_{t-1}+b_t\ ] < 2^{-b_t}. \quad (1)$$

If r is the number of bits unset by ρ then by Chebyshev's inequality we have

$$\Pr[\ r < \frac{m_{t-1}q_t}{2}\ ] \leq 4/(m_{t-1}q_t). \quad (2)$$

Since the probabilities in (1) and (2) add up to less than 1 we may choose π_t to be one of the random q_t-specializations of π_{t-1} so that neither condition holds. In this case we have $c_t \leq c_{t-1} + b_t \leq 2b_t$ and $m_t \geq \frac{m_{t-1}q_t}{2} \geq n\ 2^{-t} \prod_{i=1}^{t} q_i$ by the inductive hypothesis.
The claim follows by induction.

Now, parity requires t steps if $m_{t-1} > b_t$. This is satisfied provided

$$n\ 2^{-(t-1)} \prod_{i=1}^{t-1} q_i > b_t$$

or provided

$$n^{1-(1/3+1/9+\cdots+1/3^{t-1})} > 40^{t-1} \prod_{i=1}^{t} b_i. \quad (3)$$

The right side of (3) is of the form $3^{t^2/2}(c_1 \log_n p(n))^t$ for some constant c_1. Since $p(n) = n^{O(1)}$, $\log_n p(n)$ is a constant. Condition (3) is then satisfied for some $t = \Theta(\sqrt{\log n})$. Thus in order to compute parity we must have $T(n) = \Omega(\sqrt{\log n})$. □

Remark: It was not really necessary to require that $p(n) = n^{O(1)}$ to obtain the above lower bound. In fact one can show the same lower bound with $p(n)$ as large as $n^{2^{\epsilon\sqrt{\log n}}}$ for some $\epsilon>0$. On the other hand one does not obtain a stronger lower bound using our techniques by polynomially bounding the number of memory cells as well as the number of processors.

Using the reductions described by Chandra, Stockmeyer and Vishkin (1984) we may derive the following bounds amongst others.

Corollary 5.1: Any Abstract CRCW PRAM which sorts or adds n input bits or computes the product of two n-bit integers requires time $\Omega(\sqrt{\log n})$ if it uses a number of processors which is bounded by a polynomial in n.

6. Summary and Open Problems

The Abstract CRCW PRAM we have described seems to be a natural model in which to prove lower bounds about concurrent-read-concurrent-write machines.

It is natural to try to improve on the parity lower bound for Abstract CRCW PRAM's to match the upper bound of $\frac{\log n}{\log\log n}$ given by Chandra, Stockmeyer and Vishkin (1984). Hastad (1986) was able to do this for unbounded fan-in circuits but the processors in an Abstract CRCW PRAM are accumulating information about the input in a very different way from these circuits and for this reason it may be possible that the upper bound for these machines is not optimal.

The simulations in Stockmeyer and Vishkin (1984) show that with a restricted instruction set a CRCW PRAM with a polynomial number of processors does not need more than a polynomial number of memory cells. The bounds in Theorems 5.1 and 5.2 appear to be incomparable since they restrict different resources - one restricts processors but not cells and the other restricts cells but not processors. It seems worthwhile elaborating general relationships between the number of processors and the number of cells in the case of unrestricted instruction set machines.

The most interesting open question concerning the Abstract CRCW PRAM is the following: Are there specific Boolean functions for which we can prove stronger lower bounds than those for parity - in particular are there non-trivial lower bounds which match the values for the best known algorithms?

Acknowledgements

I thank Steve Cook for several discussions which helped initiate this work and which helped to formalize some of the main ideas in it. Also thanks to Al Borodin and Faith Fich for carefully reading drafts of this paper and for helpful comments which have corrected and improved its presentation. I am grateful to Friedholm Meyer auf der Heide for pointing me to Corollary 3.1.

References

Ajtai, M. (1983) Σ_1^1-*Formulae on Finite Structures*, Annals of Pure and Applied Logic, vol. 24, pp. 1-48.

Babai, L. (1984) private communication.

Beame, P.W. (1985) *Lower Bounds for very Powerful Parallel Machines*, manuscript.

Chandra, A.K., Stockmeyer, L.J., and Vishkin, U. (1984) *Constant Depth Reducibility*, SIAM J. Computing, vol 13(2), pp. 423-439.

Cook, S.A., Dwork, C., and Reischuk, R. (1984) *Upper and Lower Time Bounds for Parallel Random Access Machines without Simultaneous Writes*, to appear in SIAM J. Computing.

Fich, F.E., Meyer auf der Heide, F., Ragde, P., and Wigderson, A. (1985) *One, Two, Three ... Infinity: Lower Bounds for Parallel Computation*, Proc. 17th. ACM-STOC, pp. 48-58.

Furst, M., Saxe, J.B., and Sipser, M. (1984) *Parity, Circuits, and the Polynomial Time Hierarchy*, Mathematical Systems Theory, vol. 17, no. 1, pp. 13-28.

Hastad, J. (1986) *Improved Lower Bounds for Small Depth Circuits*, This proceedings.

Kucera, L. (1982) *Parallel Computation and Conflicts in Memory Access*, Information Processing Letters, vol. 14, no. 2, pp. 93-96.

Meyer auf der Heide, F. and Reischuk, R. (1984) *On the Limits to Speed Up Parallel Machines by Large Hardware and Unbounded Communication*, Proc. 25th. FOCS, pp. 56-64.

Meyer auf der Heide, F. and Wigderson, A. (1985) *The Complexity of Parallel Sorting*, Proc. 26th. IEEE-FOCS, pp. 532-540.

Parberry, I. (1985) manuscript.

Savage, J.E. (1976) *The Complexity of Computing*, Wiley.

Snir, M. (1985) *On Parallel Searching*, SIAM J. Computing, vol. 14(3), pp. 688-708.

Stockmeyer, L.J., and Vishkin, U. (1984) *Simulation of Parallel Random Access Machines by Circuits*, SIAM J. Computing, vol 13(2), pp. 404-422.

Vishkin, U., and Wigderson, A. (1985) *Trade-Offs Between Depth and Width in Parallel Computation*, SIAM J. Computing., vol 14(2), pp. 303-314.

Yao, A.C. (1985) *Separating the Polynomial-Time Hierarchy by Oracles: Part I*, Proc. 26th. IEEE-FOCS, pp. 1-10.