Optimal Bounds for Decision Problems on the CRCW PRAM

(Extended Abstract)

Paul Beame†
Johan Hastad‡

Laboratory for Computer Science
Massachusetts Institute of Technology*
545 Technology Square
Cambridge, MA 02139

Abstract

We prove optimal $\Omega(\log n/\log\log n)$ lower bounds on the time for CRCW PRAM's with polynomially bounded numbers of processors or memory cells to compute parity and a number of related problems. We also exhibit a strict time hierarchy of explicit Boolean functions of n bits on such machines which holds up to $O(\log n/\log\log n)$ time. Furthermore, we show that almost all Boolean functions of n bits require $\log n - \log\log n + \Omega(1)$ time when the number of processors is at most polynomial in n. Our bounds do not place restrictions on the uniformity of the algorithms nor on the instruction sets of the machines.

1. Introduction

One of the most widely used models of parallel computation is the parallel random access machine (PRAM). In this model any processor can access any memory location at given time-step. The most powerful form of the PRAM, the CRCW PRAM, in which both concurrent read and concurrent write accesses are allowed, has received particular attention both from designers of algorithms and from those studying the limitations of parallel machine computation. Despite the significant interest, the only non-trivial lower bounds for decision problems on CRCW PRAM's that do not have drastic restrictions placed on either their processor and memory resources or on the instruction sets of their processors are due independently to Beame [Be2] and to Li and Yesha [LY]. The lower bounds are for parity and related problems and are far from optimal. In both of these bounds no restriction is placed on the instruction set of the processors, no limitation is placed on how much information a single memory location may store, and the resources allowed are only polynomially bounded. We will call a machine with these properties an *abstract* or *ideal* PRAM.

In this very general setting we prove the first optimal bound for any non-trivial decision problem on the CRCW PRAM by showing a time lower bound of $\Omega(\log n/\log\log n)$ for parity which matches the known upper bound. This lower bound holds even in the cases when only one of the two resources, processors or memory cells, is bounded by a polynomial in the input size. Because parity constant-depth reduces to a large number of problems, this $\Omega(\log n/\log\log n)$ time lower bound for the CRCW PRAM applies to a wide variety of interesting functions which include sorting or adding n bits, as well as multiplying two n-bit integers.

Also, by looking at the so-called 'Sipser' functions which are defined by circuits, we obtain a very sharp time hierarchy for CRCW PRAM's of polynomial bounded resources. That is, for every time bound $T(n)$ at most $\log n/(3\log\log n) - \Theta(\log n/(\log\log n)^2)$ we exhibit a family of functions which is computable in time bound T with n processors and memory cells, but which cannot be computed even 2 steps faster by any machine with a polynomial bound on the number of processors. A similar sep-

† Supported by a University of Toronto Open Fellowship and by NSF grant PYI-25800.

‡ Supported by an IBM Postdoctoral Fellowship and supported in part by NSF grant DCR MCS-8509905

* Much of this research was done while the first author was at the University of Toronto.

aration holds for machines with a polynomial bound on the number of memory cells.

The proofs of both these results follow lines similar to the proofs in [Be1] and [Be2] and involve showing new lemmas which generalize the key lemmas used in Hastad's unbounded fan-in circuit lower bounds ([Ha1] and [Ha2]).

We also prove a tight $\Theta(\log n)$ lower bound on the time to compute almost all n-bit Boolean functions on CRCW PRAM's with polynomial numbers of processors.

2. History of the Problem

Much of the lower bound work for CRCW PRAM's has been based on their close relationship to unbounded fan-in circuits. These were defined by Furst, Saxe and Sipser [FSS] largely as a tool for trying to get an oracle to separate the polynomial time hierarchy from PSPACE. Stockmeyer and Vishkin [SV] showed that simple CRCW PRAM's can simulate unbounded fan-in circuits with essentially the same number of processors as the circuit size and the same time as the circuit depth. In fact, by restricting the instruction set of the CRCW PRAM to a limited set that includes addition, comparison, indirect addressing and a few related instructions, Stockmeyer and Vishkin also showed that unbounded fan-in circuits can easily simulate restricted CRCW PRAM's. The size of the resulting circuit is polynomial in the number of processors multiplied by the time and its depth is only a constant factor larger than the time. Using the latter result and a $\Omega(\log^* n)$ lower bound of [FSS] on the depth of polynomial size unbounded fan-in circuits computing parity, [SV] obtained lower bounds for this restricted form of CRCW PRAM.

Because disjunctive normal form formulas are unbounded fan-in circuits of depth two it follows that all Boolean functions may be computed in two steps using exponential resources on the CRCW PRAM. However, it is not reasonable to be using exponentially many processors and memory cells. With polynomial resource bounds, CRCW PRAM's can compute any function with formula size $n^{O(1)}$ in time $O(\log n/\log\log n)$, using an algorithm based on an upper bound of size $n^{O(1)}$ and depth $O(\log n/\log\log n)$ for unbounded fan-in circuits given by Chandra, Stockmeyer and Vishkin [CSV].

Since Stockmeyer and Vishkin's paper, the lower bounds for unbounded fan-in circuits have been significantly improved. The series of improvements ([Aj], [Ya]) culminated in the work of Hastad [Ha1] who obtained an $\Omega(\log n/\log\log n)$ depth lower bound for polynomial size circuits computing parity which matches the bound from Chandra, Stockmeyer and Vishkin's algorithm mentioned above. However, the CRCW PRAM lower bounds which follow using Stockmeyer and Vishkin's simulation are still not entirely satisfactory since the bounds rely in an essential way on the specific restriction which is placed on the instruction set. Some operations that are prohibited in this model seem to be perfectly reasonable ones.

Abstract CRCW PRAM's can be shown to be much more powerful than these restricted machines; because of their equivalence with unbounded fan-in circuits, restricted CRCW PRAM's with polynomially many processors require exponential time to compute almost all Boolean functions whereas an abstract PRAM only takes $O(\log n)$ time without even using its power of concurrent reads or writes. Nevertheless, for certain specific functions we shall see that, by using direct techniques, lower bounds as strong as those derived for these restricted CRCW machines can be obtained for the most powerful model of CRCW PRAM.

By applying and modifying the techniques of [FSS], Beame [Be1] derived the first non-trivial lower bound which applies to the CRCW PRAM model described here. He showed that any CRCW PRAM computing the parity function with $n^{O(1)}$ memory cells and an unbounded number of processors requires time $\Omega(\sqrt{\log\log n})$. Later, using the main lemma in [Ha1], Beame [Be2] obtained the following: any CRCW PRAM which computes the parity function with $n^{O(1)}$ processors (in fact with as many as $n^{2^{\delta\sqrt{\log n}}}$ processors for some $\delta > 0$) and unbounded memory requires time $\Omega(\sqrt{\log n})$. With the same techniques, an $\Omega(\sqrt{\log n})$ lower bound is easily shown for common-write CRCW PRAM's (for definitions see section 3) which have no bound on the number of processors but have a bound of $O(n^{2^{\delta\sqrt{\log n}}})$ on the number of cells for some $\delta > 0$.

It was noted by Chor [Ch] and Li and Yesha [LY] that a simulation of abstract CRCW PRAM's by unbounded fan-in circuits can be combined directly with Hastad's circuit lower bound to obtain the $\Omega(\sqrt{\log n})$ lower bound. However, this simulation does not yield the above lower bound for the common-write model with an unbounded number of processors. The simulation states that any CRCW PRAM solving a decision problem on n Boolean inputs using $p(n)$ processors and $T(n)$ time can be simulated by an unbounded fan-in circuit of size $p(n)^{2^{T(n)+O(1)}}$ and depth $O(T(n))$.

Beame [Be2] and Li and Yesha [LY] have also independently shown optimal bounds on the time needed by CRCW PRAM's to compute functions whose many-bit output is required to appear in a single memory cell. However, as was noted in [Be2], such an output requirement is somewhat artificial and the lower bounds disappear if each bit of the output is allowed to appear in a separate memory cell.

3. Definitions and Preliminaries

Definition: A *CRCW PRAM* is a shared memory machine with processors $P_1, \ldots, P_{p(n)}$ which communicate through memory cells $C_1, \ldots, C_{c(n)}$. The input is initially stored in the first n cells of memory, $C_1, \ldots, C_n$. Initially all cells other than the input cells contain the value 0. The output of the machine is the value in the cell C_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 C_j of shared memory, then, depending on the contents, (C_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 CRCW PRAM is defined to be a *common-write* machine if the values that these processors are attempting to write are always the same.

In studying the progress of CRCW PRAM computations, what is important is the set of inputs which lead to a given value in a memory cell or a given state of a processor at a particular time step. 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 a 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 C_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 C_j at the end of time step t.

We look at a measure of progress which was used in [Be1] and [Be2] to prove lower bounds for CRCW PRAM's.

Definition: Let f be a Boolean function defined on a set $I \subseteq \{0,1\}^n$ A Boolean formula F *represents* f on I if the inputs $x \in I$ satisfy F exactly when $f(x) = 1$. Let the *maximum clause length* of a DNF formula F be the maximum number of literals in any clause of F. The (*Boolean*) *degree* of f on I, $\delta(f)$, is the smallest maximum clause length of all disjunctive normal form (DNF) formulas representing f on I. We extend this definition to sets of functions $\mathcal{F}$ by letting $\delta(\mathcal{F}) = \max_{f \in \mathcal{F}} \delta(f)$.

The terminology of degree is derived from the standard way of writing a formula with the Boolean $\vee$ as addition and the Boolean $\wedge$ as multiplication and then viewing the resulting formula as a polynomial. This should not be confused with the degree of a polynomial in the finite field of two elements where the exclusive-or rather than the $\vee$ is the appropriate additive operation.

In the notation of many lower bound proofs for monotone formulae, we could define the prime implicants and prime clauses of a Boolean function f. (Prime clauses are essentially prime implicants of $\bar{f}$.) These have been described as minterms and maxterms respectively in the notation used by Yao [Ya] or Hastad [Ha1]. Observe that the degree of a function and the length of its longest minterm or maxterm may differ because its longest minterm may be longer than the longest clause in an optimal DNF formula representing it.

Definition: Let A be a partition of a set $I \subseteq \{0,1\}^n$. Define the *degree* of A, $\delta(A)$, to be $\delta(\mathcal{F}_A)$ on I where $\mathcal{F}_A$ is the set of characteristic functions of the equivalence classes of A in I.

The major proof technique of the lower bounds for parity on unbounded fan-in circuits is the use of restrictions to set some of the input bits. Using restrictions permits a simplified description of the results of computations but does not drastically reduce the difficulty of the function being computed. The main idea behind using them is that, although apparently complex operations like the OR of n bits are computed in one step, by setting relatively few inputs to 0 or 1 the results of these operations are simple. In the case of the OR of n bits, setting a single input to 1 makes it trivial.

Definition: A *restriction* π *on* $K \subseteq \{1, \ldots, n\}$ is a function $\pi : K \rightarrow \{0, 1, *\}$ where:

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

We define the results of applying a restriction π to a partition, $A\lceil_\pi$, a function, $f\lceil_\pi$, and a Boolean formula, $F\lceil_\pi$, in the natural way. If σ and τ are restrictions then $\sigma\tau$ is a restriction which is the result of applying σ first and then applying τ. For any $K \subseteq \{1,\ldots,n\}$ define $Proj[K]$ to be the set of restrictions which assign 0 or 1 exactly to the inputs in K.

In several places we will need the following simple observation.

Lemma 3.1: *Let A be a partition of a cube $I \subseteq \{0,1\}^n$. For every $K \subseteq \{1,\ldots,n\}$ there exists a restriction $\sigma \in Proj[K]$ such that $\delta(A) \leq |K| + \delta(A\lceil_\sigma)$.*

The hard part in showing that restrictions simplify the results of CRCW PRAM computations is naturally the very powerful concurrent write operation since the read operation is simply the interaction of individual processors with single cells. It will be useful to define an abstraction of this operation in order to be able to describe conveniently the actions of restrictions on the new cell partitions which result from the concurrent writes. It also will turn out that, in describing the effects of restrictions on the processor partitions, we use a special case of this abstraction.

Definition: We say that an input $x \in \{0,1\}^n$ *satisfies* a Boolean function $F : \{0,1\}^n \rightarrow \{0,1\}$ if $F(x) = 1$. We say that x *falsifies* F if $F(x) = 0$.

Definition: A *graded set of Boolean functions* is a set $\mathcal{G}$ of Boolean functions such that each $F \in \mathcal{G}$ has an associated positive integer *grade*, $\gamma(F)$ (or has grade $= \infty$) and no two functions of a given grade are simultaneously satisfiable.

Definition: For any graded set of Boolean functions, $\mathcal{G}$, *the partition determined by* $\mathcal{G}$, $\langle\mathcal{G}\rangle$, on $\{0,1\}^n$ is the partition such that $x, y \in \{0,1\}^n$ are in the same equivalence class if and only if:

(a) x and y both satisfy some function $F \in \mathcal{G}$, and x and y both falsify all $F' \in \mathcal{G}$ with $\gamma(F') < \gamma(F)$.

or (b) x and y both falsify all functions $F \in \mathcal{G}$.

The reflexivity and symmetry of the relation above are obvious. The transitivity is a simple consequence of the fact that the definition of a graded set of functions excludes the possibility that two functions of a given grade are simultaneously satisfiable. For technical reasons the following straightforward lemma is convenient.

Lemma 3.2: *Let $\mathcal{G}$ be a graded set of Boolean functions. If π is a restriction then $\langle\mathcal{G}\rangle\lceil_\pi$ is the same partition as $\langle\mathcal{G}\lceil_\pi\rangle$ on $\{0,1\}^n\lceil_\pi$.*

We note that the above definitions can be carried over easily to Boolean formulas which *represent* the Boolean functions in the obvious way. Observe that if $\mathcal{F}$ represents $\mathcal{G}$ on $\{0,1\}^n\lceil_\pi$ then $\langle\mathcal{F}\rangle\lceil_\pi = \langle\mathcal{G}\rangle\lceil_\pi$. Also, the notion of degree applies to graded sets of Boolean functions simply using the natural definition of degree for sets of functions. It is easy to see that a graded set of Boolean functions $\mathcal{G}$ can be represented on a cube $\{0,1\}^n\lceil_\pi$ by a graded set of DNF formulas $\mathcal{F}$, each with maximum clause length bounded by $\delta(\mathcal{G}\lceil_\pi)$.

Definition: Let M be a CRCW PRAM. Define $\mathcal{G}(M,j,t)$ to be the graded set of Boolean functions as follows:

(i) For each positive integer i, the functions of grade i in $\mathcal{G}(M,j,t)$ are the characteristic functions of those equivalence classes in $P(M,i,t)$ on which P_i writes into cell C_j during time step t.

(ii) The functions of grade ∞ in $\mathcal{G}(M,j,t)$ are all the characteristic functions of the equivalence classes in $C(M,j,t-1)$.

Lemma 3.3: *Let M be a CRCW PRAM. $\langle\mathcal{G}(M,j,t)\rangle$ is a refinement of $C(M,j,t)$ on $\{0,1\}^n$.*

Proof: The way in which a partition is determined by a graded set of functions imitates the priority write operation of the CRCW PRAM. Condition (b) in the definition of the partition determined by a graded set of function cannot occur here since every input satisfies the characteristic function of some equivalence class in $C(M,j,t-1)$. Condition (a) in this definition corresponds to one of two cases. Either the input causes processor P_i to write and P_i is guaranteed to succeed since no lower numbered processor attempts to write, or no processor writes and thus the previous value in the cell remains (we view this as the cell writing its old value back to itself). □

The general method we employ for showing lower bounds on CRCW PRAM computations for decision problems is as follows. We show that after certain restrictions (which set more inputs as time progresses) are applied to the inputs, the processor and cell partitions have only small degree relative to the degree required to solve the problems. In using restrictions to obtain our lower bounds we must maintain a balance between the amount of simplification that a restriction achieves and the number of inputs it sets.

4. Tight Lower Bounds for Parity

We now state our main result.

Theorem 4.1: *If M is a CRCW PRAM which computes the parity function in time $T = T(n)$ then for sufficiently large n*

(a) the total hardware $h(n) = p(n) + c(n)$ must be at least $2^{[\frac{1}{24}n^{1/T}-2]}$,

(b) the number of processors $p(n)$ must be at least $2^{[\frac{1}{96}n^{1/T}-2]}$ even if the number of memory cells is infinite, and

(c) the number of memory cells $c(n)$ must be at least $2^{[\frac{1}{12}(n/T!)^{1/T}-2]}$ even if the number of processors is infinite.

For the proofs of each of the parts of this theorem we define restrictions π_t for each step t of the computation such that after step t and after π_t is applied, the cell (and processor) partitions all have degree less than the number of unset variables. The lower bound follows since setting variables of parity just leaves a smaller parity function and any representation of parity in DNF has clauses which depend on *all* the unset variables.

In order to prove the existence of restrictions that satisfy these properties we need an appropriate probability space from which to choose restrictions. This distribution was introduced by Furst, Saxe, and Sipser [FSS] and has been used in several subsequent lower bound proofs for unbounded fan-in circuits.

Definition: Let $K \subseteq \{1,\ldots,n\}$. Define R_p^K to be a probability space of restrictions on K where for a random ρ chosen from R_p^K, independently for each $i \in K$, $\rho(i)$ is $*$ with probability p and $\rho(i)$ is 0 or 1 with equal probability $(1-p)/2$.

The outline above is now carried out by proving two lemmas. The first tells us that many variables remain unset and the second tells us that the degrees of the partitions do not increase.

Lemma 4.1: *Let $L \subseteq \{1,\ldots,n\}$ and $0 < p < 1$ such that $p(1-p)|L|$ is at least m_0 for some absolute constant m_0. Choose ρ at random from R_p^L. The probability that ρ leaves at least $p|L|$ inputs unset is greater than 1/3.*

Lemma 4.2: *Let M be a CRCW PRAM just prior to a read or write operation, all of whose processor and cell partitions have degree at most $r \geq 1$ with variables from $\{x_i\}_{i\in L}$. Let A be either a new processor partition resulting from a concurrent read of M or a new cell partition resulting from a concurrent write of M. Choose ρ at random from R_p^L. For $s > 0$ we have*

$$\Pr[\,\delta(A\lceil_\rho) \geq s\,] < (6pr)^s.$$

This is an easy corollary of Lemma 3.3 and the following lemma which is the key generalization of the main lemma of Hastad [Ha1].

Lemma 4.3: *Let $\mathcal{G}$ be a graded set of DNF formulas on inputs $\{x_i\}_{i\in L}$ with maximum clause length bounded by $r \geq 1$ where $L \subseteq \{1,\ldots,n\}$. Let F be an arbitrary function on $\{0,1\}^n$. Let ρ be a random restriction chosen from R_p^L. Then, if $\langle \mathcal{G}\lceil_\rho\rangle$ is the partition determined by $\mathcal{G}\lceil_\rho$, for $s \geq 0$ we have*

$$\Pr[\,\delta(\langle \mathcal{G}\lceil_\rho\rangle) \geq s \mid F\lceil_\rho = 0\,] \leq \beta^s$$

where $\beta > 0$ satisfies

$$\left(\frac{4p}{\beta(1+p)} + 1\right)^r = 2.$$

Proof: We first note that we only need to consider finite graded sets of formulas (i.e. $|\mathcal{G}|$ is finite). This follows since there are only a finite number of different input strings and so only a finite number of ways in which some formula in $\mathcal{G}$ can be satisfied and all smaller ones falsified. Also, it is trivial to see that the lemma holds for $s = 0$ or $\beta \geq 1$ so we can assume that $s > 0$ and $\beta < 1$.

The rest of the proof proceeds by induction on the total number of clauses in the formulas in $\mathcal{G}$. The intuitive idea is that as we work along the clauses one by one: if ρ falsifies a clause, then we are left with essentially the same problem as before; if ρ does not then, given the fact that it does not, it is much more likely that ρ satisfies the clause and ensures that the remaining partition has only one class than that ρ leaves any input in the clause unset.

In this proof for readability we will write $\delta\langle\mathcal{G}\rangle$ instead of $\delta(\langle\mathcal{G}\rangle)$.

BASE CASE: There are no clauses in the formulas in $\mathcal{G}$. In this case the formulas are all identically 0 and so all inputs are equivalent with respect to $\mathcal{G}$. Thus the partition determined by $\mathcal{G}\lceil_\rho$ consists of a single class so $\delta\langle\mathcal{G}\lceil_\rho\rangle = 0$ and the lemma holds for $\mathcal{G}$.

INDUCTION STEP: Assume that the lemma holds for all graded sets of formulas $\mathcal{G}'$ with fewer clauses than the formulas of $\mathcal{G}$. Let F_1 be a formula in $\mathcal{G}$ which has lowest grade among those formulas in $\mathcal{G}$ which

are not identically 0; let C_1 be a clause of F_1. We can analyse the probability by considering separately the cases in which ρ does or does not force clause C_1 to be 0. The failure probability, the probability that $\delta\langle\mathcal{G}\lceil_\rho\rangle \geq s$, is an average of the failure probabilities in these two cases. Thus

$$\Pr[\,\delta\langle\mathcal{G}\lceil_\rho\rangle \geq s \mid F\lceil_\rho = 0\,] \leq$$
$$\max(\,\Pr[\,\delta\langle\mathcal{G}\lceil_\rho\rangle \geq s \mid F\lceil_\rho = 0 \wedge C_1\lceil_\rho = 0\,],$$
$$\Pr[\,\delta\langle\mathcal{G}\lceil_\rho\rangle \geq s \mid F\lceil_\rho = 0 \wedge C_1\lceil_\rho \neq 0\,]\,).$$

The first term in the maximum is $\Pr[\,\delta\langle\mathcal{G}\lceil_\rho\rangle \geq s \mid (F \vee C_1)\lceil_\rho = 0\,]$. Let $\tilde{F}_1$ be F_1 with clause C_1 removed; thus $F_1 = C_1 \vee \tilde{F}_1$ and $\tilde{F}_1 \neq F_1$. Let $\tilde{\mathcal{G}}$ be the same as $\mathcal{G}$ with formula F_1 replaced by $\tilde{F}_1$. In this case $C_1\lceil_\rho = 0$ so $F_1\lceil_\rho = \tilde{F}_1\lceil_\rho$ and thus $\langle\mathcal{G}\lceil_\rho\rangle = \langle\tilde{\mathcal{G}}\lceil_\rho\rangle$. In other words, when $C_1\lceil_\rho = 0$, the lemma requires a bound on $\Pr[\,\delta\langle\tilde{\mathcal{G}}\lceil_\rho\rangle \geq s \mid (F \vee C_1)\lceil_\rho = 0\,]$. Since $\tilde{\mathcal{G}}$ has one fewer clause than $\mathcal{G}$ does, the inductive hypothesis implies that this probability is at most β^s.

The estimation of the second term in the maximum is more difficult. Let $T \subseteq L$ be the set of variables appearing in clause C_1. By hypothesis $|T| \leq r$. Let ρ_T be the restriction of ρ to the variables in T. The condition that $C_1\lceil_\rho \neq 0$ is equivalent to the condition that $C_1\lceil_{\rho_T} \neq 0$. We analyse the cases based on the subset Y of the variables in T to which ρ_T assigns $*$; we use the notation $*(\rho_T) = Y$ to denote the event that the variables in T which are assigned $*$ by ρ_T are exactly those in Y. Then

$$\Pr[\,\delta\langle\mathcal{G}\lceil_\rho\rangle \geq s \mid F\lceil_\rho = 0 \wedge C_1\lceil_{\rho_T} \neq 0\,]$$
$$= \sum_{Y \subseteq T} \Pr[\,\delta\langle\mathcal{G}\lceil_\rho\rangle \geq s \wedge *(\rho_T) = Y \mid F\lceil_\rho = 0 \wedge C_1\lceil_{\rho_T} \neq 0\,]. \quad (1)$$

Consider the case in which $Y = \phi$. Then ρ_T sets every variable in T so the value of C_1 is forced by ρ_T. But since we already know that $C_1\lceil_{\rho_T} \neq 0$ we must have $C_1\lceil_{\rho_T} = 1$. In this case every input satisfies $F_1\lceil_\rho$ and since F_1 has lowest grade we know that all inputs are equivalent with respect to the $\langle\mathcal{G}\lceil_\rho\rangle$. It follows that $\delta\langle\mathcal{G}\lceil_\rho\rangle = 0$ so the term corresponding to $Y = \phi$ has probability 0. The sum in (1) then becomes

$$\Pr[\,\delta\langle\mathcal{G}\lceil_\rho\rangle \geq s \mid F\lceil_\rho = 0 \wedge C_1\lceil_{\rho_T} \neq 0\,]$$
$$= \sum_{Y \subseteq T, Y \neq \phi} \Pr[\,\delta\langle\mathcal{G}\lceil_\rho\rangle \geq s \wedge *(\rho_T) = Y \mid F\lceil_\rho = 0 \wedge C_1\lceil_{\rho_T} \neq 0\,]$$
$$= \sum_{Y \subseteq T, Y \neq \phi} \Pr[\,\delta\langle\mathcal{G}\lceil_\rho\rangle \geq s \mid F\lceil_\rho = 0 \wedge C_1\lceil_{\rho_T} \neq 0 \wedge *(\rho_T) = Y\,] \times \Pr[\,*(\rho_T) = Y \mid F\lceil_\rho = 0 \wedge C_1\lceil_{\rho_T} \neq 0\,] \quad (2)$$

by simple conditional probability.

We tackle the latter term in each of these products first. If we let $\rho_T(Y) = *$ denote the event that every variable in Y is unset by ρ_T then elementary probability yields

$$\Pr[\,*(\rho_T) = Y \mid F\lceil_\rho = 0 \wedge C_1\lceil_{\rho_T} \neq 0\,]$$
$$\leq \Pr[\,\rho_T(Y) = * \mid F\lceil_\rho = 0 \wedge C_1\lceil_{\rho_T} \neq 0\,].$$

Then as in [Ha1] we have

$$\Pr[\,\rho_T(Y) = * \mid F\lceil_\rho = 0 \wedge C_1\lceil_{\rho_T} \neq 0\,] \leq (\frac{2p}{1+p})^{|Y|}.$$

Now we look at the first term in each product in (2). The condition that $C_1\lceil_{\rho_T} \neq 0 \wedge *(\rho_T) = Y$ exactly specifies $\rho_T = \rho\lceil_T$ since it means that every variable in $T \setminus Y$ is set to 0 or 1 in the way which does not force the value of C_1 to 0 and that every variable in Y is set to $*$. We let F' be $F \vee G$ where $G\lceil_\rho = 0$ if and only if ρ sets the variables in $T \setminus Y$ in the unique way that does not force clause C_1 to 0. Thus

$$\Pr[\,\delta\langle\mathcal{G}\lceil_\rho\rangle \geq s \mid F\lceil_\rho = 0 \wedge C_1\lceil_{\rho_T} \neq 0 \wedge *(\rho_T) = Y\,]$$
$$= \Pr[\,\delta\langle\mathcal{G}\lceil_\rho\rangle \geq s \mid F'\lceil_\rho = 0 \wedge *(\rho_T) = Y\,].$$

Now, the condition $*(\rho_T) = Y$ means that the variables in Y are unset by ρ and that the variables in $T \setminus Y$ are all set by Y. The latter part of this condition is implied by the condition $F'\lceil_\rho = 0$. Thus we do not change the events by rewriting the probability as

$$\Pr[\,\delta\langle\mathcal{G}\lceil_\rho\rangle \geq s \mid F'\lceil_\rho = 0 \wedge *(\rho_Y) = Y\,]$$

where ρ_Y is ρ restricted to the variables in Y. The condition $*(\rho_Y) = Y$ means that every variable in Y is unset by ρ.

If $|Y| \leq s$ then, by Lemma 3.1,

$$\Pr[\,\delta\langle\mathcal{G}\lceil_\rho\rangle \geq s \mid F'\lceil_\rho = 0 \wedge *(\rho_Y) = Y\,]$$
$$\leq \Pr[\,\exists \sigma \in Proj[Y], \delta\langle(\mathcal{G}\lceil_\sigma)\lceil_\rho\rangle \geq s - |Y| \mid F'\lceil_\rho = 0 \wedge *(\rho_Y) = Y\,]$$
$$\leq \sum_{\sigma \in Proj[Y]} \Pr[\,\delta\langle(\mathcal{G}\lceil_\sigma)\lceil_\rho\rangle \geq s - |Y| \mid F'\lceil_\rho = 0 \wedge *(\rho_Y) = Y\,]$$
$$= \sum_{\sigma \in Proj[Y]} \Pr[\,\delta\langle(\mathcal{G}\lceil_\sigma)\lceil_{\rho'}\rangle \geq s - |Y| \mid F'\lceil_{\rho'} = 0 \wedge *(\rho_Y) = Y\,] \quad (3)$$

where ρ' is the restriction of ρ to the set $L' = L \setminus Y$. This last equality holds because ρ' sets exactly the same inputs that ρ does.

Because the probabilities on L' are independent of those on Y, the condition on ρ_Y does not affect the probabilities for ρ' so it can be eliminated without changing the probabilities in (3). Furthermore, because the probabilities on L' for ρ chosen at random from R_p^L are the same as those for a ρ' chosen from $R_p^{L'}$, the sum in (3) is equivalent to

$$\sum_{\sigma \in Proj[Y]} \Pr[\, \delta\langle(\mathcal{G}\lceil_\sigma)\lceil_{\rho'}\rangle \geq s - |Y| \mid F'\lceil_{\rho'} = 0\,] \quad (4)$$

where ρ' is a restriction chosen at random from $R_q^{L'}$

Because of the fact that σ sets all inputs in Y and $F'\lceil_{\rho'} = 0$ we know that $\sigma\rho'$ sets all the inputs in T and thus forces the value of C_1. If $C_1\lceil_{\sigma\rho'} = 1$ then all inputs in $\langle(\mathcal{G}\lceil_\sigma)\lceil_{\rho'}\rangle$ are equivalent and thus $\delta\langle(\mathcal{G}\lceil_\sigma)\lceil_\rho\rangle = 0 \leq |Y| - s$. Otherwise $C_1\lceil_{\sigma\rho'} = 0$ and then $\langle(\mathcal{G}\lceil_\sigma)\lceil_{\rho'}\rangle = \langle(\tilde{\mathcal{G}}\lceil_\sigma)\lceil_{\rho'}\rangle$ since $\tilde{F}_1\lceil_{\sigma\rho'} = F_1\lceil_{\sigma\rho'}$. Thus the sum in (4) is equivalent to

$$\sum_{\sigma \in Proj[Y]} \Pr[\, \delta\langle(\tilde{\mathcal{G}}\lceil_\sigma)\lceil_{\rho'}\rangle \geq s - |Y| \mid F'\lceil_{\rho'} = 0\,].$$

Because $\tilde{\mathcal{G}}\lceil_{\sigma_1}$ has strictly fewer clauses than $\mathcal{G}$ and because it only has input variables from L' we can apply the inductive hypothesis to bound the probabilities in each term in this sum by $\beta^{s-|Y|}$. For each Y the number of terms in the above sum is at most $|Proj[Y]| = 2^{|Y|}$ so we obtain a total bound of $2^{|Y|}\beta^{s-|Y|}$.

If $|Y| > s$ then we simply make the pessimistic assumption of failure, i.e. that the degree of the resulting partition is too large. Since $\beta < 1$ and $s - |Y| < 0$ we certainly have $1 < 2^{|Y|}\beta^{s-|Y|}$. Thus

$$\Pr[\, \delta\langle\mathcal{G}\lceil_\rho\rangle \geq s \mid F\lceil_\rho = 0 \wedge C_1\lceil_{\rho_T} \neq 0 \wedge *(\rho_T) = Y\,]$$

is at most $2^{|Y|}\beta^{s-|Y|}$.

Finally, substituting these bounds in (2) we obtain a total failure probability of at most

$$\begin{aligned}
&\sum_{Y \subseteq T, Y \neq \phi} \left(\frac{2p}{1+p}\right)^{|Y|} 2^{|Y|}\beta^{s-|Y|} \\
&\quad = \beta^s \sum_{i=1}^{|T|} \binom{|T|}{i} \left[\frac{4p}{\beta(1+p)}\right]^i \\
&\quad = \beta^s \left[\left(\frac{4p}{\beta(1+p)} + 1\right)^{|T|} - 1\right] \\
&\quad \leq \beta^s \left[\left(\frac{4p}{\beta(1+p)} + 1\right)^{r} - 1\right] \\
&\quad = \beta^s
\end{aligned}$$

using the definition of β. Thus the lemma holds for $\mathcal{G}$ and by induction we have proved the lemma. □

Proof of Theorem 4.1 (sketch): For part (a), we show that the degree of the processor and cell partitions can be maintained at $\log 4h(n)$ by setting all but a fraction of at least $p = 1/\log 4h(n)$ of the remaining input bits. Lemmas 4.1 and 4.2 imply that a restriction chosen randomly from R_p will almost certainly work in a given time step for each memory cell and processor. We then must check that the probability of failure at each step, which is bounded by the sum of the failure probabilities in each of the $h(n)$ processors and memory cells, is < 1. In order to have computed parity in T steps the degree must be equal to the number of unset bits and the bound follows.

The bound in part (b) follows by similar reasoning and the observation that at each step if the degree of a processor is at most s then it only has the potential of writing into at most 2^s different memory cells. Thus the number of memory cells which have to be considered at each step is bounded.

Finally, for part (c) we maintain different bounds on the degree of the processor and cell partitions. The degree of the cell partitions is still maintained at $s = \log 4c(n)$, however the degree of the processor partitions can only be maintained at st for step t of the algorithm. This means that many more input bits must be set at each step in order to keep down the degree of the result of a write operation. For further details see [Be3]. □

We can restate the resource trade-offs given in Theorem 4.1 in terms of the time required by practically sized CRCW PRAM's to compute the parity function:

Corollary 4.1: *If M is a CRCW PRAM which computes the parity function in time $T = T(n)$ then*

(a) if the number of processors $p(n) = n^{O(1)}$ then

$$T(n) \geq \frac{\log n}{\log\log n} - O\left(\frac{\log n}{(\log\log n)^2}\right),$$

even if the number of memory cells is infinite

(b) if the number of memory cells $c(n) = n^{O(1)}$ then

$$T(n) \geq \frac{\log n}{2\log\log n} - O\left(\frac{\log n}{(\log\log n)^2}\right),$$

even if the number of processors is infinite.

A close look at the algorithm given by Chandra, Stockmeyer, and Vishkin [CSV] for computing functions with polynomial formula size, shows that parity can be computed by CRCW PRAM's with polynomially many processors and memory cells in time $\log n/\log\log n - c\log n/(\log\log n)^2$, where the constant c depends on the exponent in the polynomial bound on the number of processors and cells. The only difference between our bound (a) and this one is that this constant c is smaller relative to the exponent of the polynomial which bounds the number of processors and cells than is the constant in our lower bound.

Using the constant-depth reductions given in [FSS] and [CSV], these same tight lower bounds for parity can be obtained for a large number of functions. We assume that the reader is familiar with the definitions of most of these problems; the terminology is from [CSV].

Corollary 4.2:[FSS], [CSV] *IF M is a CRCW PRAM computing any of the following decision problems, the bounds in Corollary 4.1 hold:*
THRESHOLD, MAJORITY, UNDIRECTED GRAPH CONNECTIVITY, UNDIRECTED CYCLE DETECTION IN GRAPHS, BI-PARTITE MATCHING, CIRCUIT VALUE PROBLEM
The bounds in Corollary 4.1 also hold for computing all the bits of the following function problems:
MULTIPLICATION, SORTING, BIT SORTING, MULTIPLE-ADDITION, BIT SUM, NETWORK FLOW WITH UNARY CAPACITIES

The MULTIPLE-ADDITION problem is just the integer addition problem discussed in [Be2] and [LY]. This corollary shows that when the output is permitted to be represented as bits, the time complexity is $\Theta(\log n/\log\log n)$ for machines with polynomially bounded hardware. This complements the previous results which showed that, when the output is required to be in a single cell, the time complexity is $\Theta(\log n)$ for such machines.

The functions listed in this corollary are by no means all the natural functions to which our parity lower bound applies but merely a representative sample of the variety of problems involved.

5. The Sipser Functions and a CRCW Time Hierarchy

In [Si], Sipser defined a set of functions F_k^m on m^k inputs for $k \geq 2$ which are described by alternating unbounded fan-in circuits of depth k and size $O(m^k)$. He obtained a strict hierarchy of polynomial-size unbounded fan-in circuits by showing that these functions required more than polynomial size circuits of depth $k-1$. Sipser's function F_k^m was described by an alternating tree of depth k of $\wedge$ and $\vee$ gates with an $\wedge$ at the root, with fan-in m at every level, and with distinct inputs at every leaf. We modify it somewhat by defining f_k^m to be a function having fan-in $a_k = \left\lceil\sqrt{\frac{1}{2}mk\log m}\right\rceil$ from the leaves, fan-in $\left\lceil\sqrt{2m/\log m}\right\rceil$ at the root and fan-in m everywhere else. The resulting function has $n = m^{k-2}\left\lceil\sqrt{2m/\log m}\right\rceil\left\lceil\sqrt{\frac{1}{2}mk\log m}\right\rceil < 4\sqrt{k}m^{k-1}$ inputs in total.

Theorem 5.1: *If M is a CRCW PRAM which computes the function f_T^m of n inputs in time $T-2$ then for m sufficiently large*

(a) the total hardware $h(n) = p(n) + c(n)$ must be at least $2^{[\frac{1}{24}(n^{1/2T}/\sqrt{2\log n})-2]}$,

(b) the number of processors $p(n)$ must be at least $2^{[\frac{1}{96}(n^{1/2T}/\sqrt{2\log n})-2]}$ even if the number of memory cells is infinite, and

(c) the number of memory cells $c(n)$ must be at least $2^{[\frac{1}{12T}(n^{1/2T}/\sqrt{2\log n})-2]}$ even if the number of processors is infinite.

Because of space considerations we merely outline the argument. Its general pattern is similar to the proofs for parity. For further details of the proofs see [Be3].

To prove this theorem we define restrictions π_t for each step t of the computation just as we did for parity such that after step t and after π_t is applied, the cell (and processor) partitions all have degree less than m and yet the function to be computed is f_{T-t}^m.

Parity is a very nice function which treats 0 and 1 equally, so it is possible to use restrictions from the probability distribution R_p^L and leave the parity function unchanged in character. The functions f_k^m which we have just described treat 0 and 1 very differently, depending upon whether k is even or odd. Also, they are not symmetric so that treating all variables equally and independently as R_p^L does is inappropriate. The functions f_k^m do have some symmetry: inputs which appear at leaves that are joined to the same bottom level gate are symmetric with respect to each other. (We call such a set of inputs a *block*.) Also, blocks which fan in to the same second level gate are symmetric with respect to each other. These sym-

metries motivated the following restrictions of Hastad [Ha2]:

Definition: Let $L \subseteq \{1,\ldots,n\}$ and let $\mathcal{L} = \{L_i\}_{i=1}^{l}$ be a partition of L into blocks. Define $R^{+}_{q,\mathcal{L}}$ to be a probability space of restrictions on L where for a random ρ chosen from $R^{+}_{q,\mathcal{L}}$ and independently for every $i \in \{1,\ldots,l\}$,
1. A parameter s_i is chosen such that $\Pr[\, s_i = * \,] = q$ and $\Pr[\, s_i = 0 \,] = 1-q$
2. Independently for each $j \in L_i$, $\Pr[\, \rho(j) = s_i \,] = q$ and $\Pr[\, \rho(j) = 1 \,] = 1-q$
Similarly $R^{-}_{q,\mathcal{L}}$ is a probability space of restrictions defined as above except that the positions of 1 and 0 are reversed.

Note that restrictions from $R^{+}_{q,\mathcal{L}}$ never assign $*$ and 0 to inputs from the same block and restrictions from $R^{-}_{q,\mathcal{L}}$ never assign $*$ and 1 to the same block. The restrictions from $R^{+}_{q,\mathcal{L}}$ are likely to set most inputs to 1 and are used for f^m_k when the bottom level gates are $\wedge$; the restrictions from $R^{-}_{q,\mathcal{L}}$ are likely to set most inputs to 0 and are used for f^m_k when the bottom level gates are $\vee$.

Definition: For a restriction ρ chosen from $R^{+}_{q,\mathcal{L}}$ let $g^{+}(\rho)$ be the restriction which agrees with ρ everywhere ρ sets inputs and which assigns 1 to all but the variable of least index in each block which is given a $*$ by ρ. Similarly, for a restriction ρ chosen from $R^{-}_{q,\mathcal{L}}$ let $g^{-}(\rho)$ be the restriction which agrees with ρ everywhere ρ sets inputs and which assigns 0 to all but the variable of least index in each block which is given a $*$ by ρ.

The definitions of g^{+} and g^{-} are intended to be cleaned up versions of the original restrictions. The idea of this lower bound is that when f^m_k has ρ applied to it, there is a copy of f^m_{k-1} sitting inside it. In this process most of the bottom level gates will end up with more than one variable and to keep degrees small while retaining the copy of f^m_{k-1} it will be essential to apply the g^{+} and g^{-} to reduce these to one variable.

As in the case of the parity function we need two lemmas, one making sure that the function we are trying to compute remains complicated after a restriction and one which controls the degree of the partitions.

Lemma 5.1: *Let $\frac{1}{4} > q \geq \sqrt{\frac{2k\log m}{m}}$, and let $\mathcal{L} = \{L_i\}_{i=1}^{l}$ be the partition of the input set of f^m_k into blocks which are the sets of inputs which fan into each of its bottom level gates.*

(i) If k is odd then, for ρ chosen at random from $R^{+}_{q,\mathcal{L}}$, the circuit that defines $f^m_k\lceil_{g^{+}(\rho)}$ contains a circuit that defines f^m_{k-1} with probability at least 2/3 for all $m > m_1$ where m_1 is a constant independent of k and q.

(ii) If k is even then, for ρ chosen at random from $R^{-}_{q,\mathcal{L}}$, the circuit that defines $f^m_k\lceil_{g^{-}(\rho)}$ contains a circuit that defines f^m_{k-1} with probability at least 2/3 for all $m > m_1$ where m_1 is a constant independent of k and q.

Lemma 5.2: *Let M be a CRCW PRAM just prior to a read or write operation, all of whose processor and cell partitions have degree at most $r \geq 1$ with variables from $\{x_i\}_{i\in L}$. Let A be either a new processor partition resulting from a concurrent read of M or a new cell partition resulting from a concurrent write of M. Let $\mathcal{L} = \{L_i\}_{i=1}^{l}$ be a partition of L. Choose ρ at random from $R^{+}_{q,\mathcal{L}}$. For $s > 0$ we have*

$$\Pr[\, \delta(A\lceil_{g^{+}(\rho)}) \geq s \,] < (6qr)^s.$$

The same result holds if $+$ replaces $-$ throughout.

As it stands, our functions f^m_T are defined only for certain numbers of inputs depending on T and m; call this number $\nu_{T,m}$. Let T be a function of n. We extend our functions to all numbers of inputs by defining $f_{T(\cdot)}$ on n inputs to be $f^m_{T(n)}$ computed on the first $\nu_{T(n),m}$ inputs, where m is the largest index such that $\nu_{T(n),m} \leq n$. We now can restate the resources required to reduce the time for computing $f_{T(\cdot)}$ on machines with reasonable resource bounds.

Corollary 5.1:

(a) For any function T such that

$$T(n) = \frac{\log n}{3\log\log n} - \omega\left(\frac{\log n}{(\log\log n)^2}\right)$$

there is a function $f_{T(\cdot)}$ of n inputs which can be computed on a CRCW PRAM with n processors and memory cells in time $T(n)$ but cannot be computed by any CRCW PRAM with a polynomially bounded number of processors, $p(n) = n^{O(1)}$, running in time $T(n) - 2$.

(b) For any function T such that

$$T(n) = \frac{\log n}{5\log\log n} - \omega\left(\frac{\log n}{(\log\log n)^2}\right)$$

there is a function $f_{T(\cdot)}$ of n inputs which can be computed on a CRCW PRAM with n processors and memory cells in time $T(n)$ but any CRCW

PRAM computing it in time $T(n)-2$ requires both the number of memory cells and the number of processors to exceed any polynomial in n.

This implies that the class of functions which can be computed in time bound $T(\cdot)-2$ on machines with reasonable resource bounds is strictly contained in the class of functions which can be computed in time $T(\cdot)$. This yields a strict time hierarchy among CRCW PRAM's.

6. Almost all Boolean functions

We can get larger lower bounds on the time complexity of Boolean functions than those in the previous sections by considering the class of almost all Boolean functions.

Lemma 6.1: *Almost all Boolean functions require unbounded fan-in circuit size $\Omega(2^{n/2})$.*

Proof: To see this, use the following argument due to Ruzzo [Ru]: Without loss of generality the negations can be pushed to the inputs by De Morgan's laws so we assume free access to inputs and their negations. The number of unbounded fan-in circuits with s gates is then just $2^{s(s+2n+1)}$ since each gate can be described by its operation (either $\wedge$ or $\vee$) and by the subset of the inputs and gates to which it is attached. Since there are 2^{2^n} Boolean functions of n inputs, it easy to see that most functions require size $\Omega(2^{n/2})$.□

Using the simulation of CRCW PRAM's by circuits given by [LY] and [Ch] (cf. §2) along with this lemma yields:

Theorem 6.1: *Almost all Boolean functions of n inputs require time $\log n - \log\log p(n) + \Omega(1)$ on a CRCW PRAM with $p(n)$ processors.*

Proof: Substituting directly in the simulation we see that any CRCW PRAM taking at most $\log n - \log\log p(n) - \omega(1)$ time can be simulated by an unbounded fan-in circuit of size $o(2^{n/2})$. But by Lemma 6.1, almost all Boolean functions of n inputs require unbounded fan-in circuits of size $\Omega(2^{n/2})$. The theorem follows immediately. □

Because of the upper bound in [Be2] of $\log n - \log\log[p(n)/n] + O(1)$ for computing any function on Boolean inputs, this bound is nearly optimal. This optimality suggests that no general improvement in the simulation of CRCW PRAM's by unbounded fan-in circuits is likely to be obtained.

7. Further Research

The $\Omega(\log n/\log\log n)$ time lower bounds for computing specific Boolean functions given in sections 4 and 5 are tantalizingly close to the $\log n - \log\log n + \Theta(1)$ time bounds for almost all Boolean functions on C?CW PRAM's with a polynomial number of processors. However, finding a specific problem in NP for which we can close this gap appears to be a formidable though fundamentally interesting task. This is because the work of Chandra, Stockmeyer and Vishkin ([SV], [CSV]) implies that such a problem would not be in NC^1 (would not have $O(\log n)$ depth combinational circuits).

References

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

[Be1] Beame, P.W. *Lower Bounds for very Powerful Parallel Machines*, manuscript, 1985.

[Be2] Beame, P.W. *Limits on the Power of Concurrent-Write Parallel Machines*, Proc. 18th ACM-STOC 1986, pp. 169-176.

[Be3] Beame, P.W. *Lower Bounds in Parallel Machine Computation*, Ph.D. Thesis, University of Toronto, 1986.

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

[Ch] Chor, B. private communication, 1986.

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

[Ha1] Hastad, J. *Improved Lower Bounds for Small Depth Circuits*, Proc. 18th ACM-STOC 1986, pp. 6-20.

[Ha2] Hastad, J. *Computational Limitations for Small Depth Circuits*, Ph.D. Thesis, M.I.T., 1986.

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

[LY] Li, M., and Yesha, Y. *New Lower Bounds for Parallel Computation*, Proc. 18th ACM-STOC 1986, pp. 177-187.

[Ru] Ruzzo, W.L. private communication, 1986.

[Si] Sipser, M. *Borel Sets and Circuit Complexity*, Proc. 15th ACM-STOC 1983, pp. 61-69.

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

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