ERCW PRAMs and Optical Communication

Philip D. MacKenzie* and Vijaya Ramachandran**

Dept. of Computer Sciences, University of Texas, Austin TX 78712-1188, USA

Abstract. This paper presents algorithms and lower bounds for several fundamental problems on the ERCW PRAM and some results for unbounded fan-in, bounded fan-out (or 'BFO') circuits. Our results for these two models are of importance because of the close relationship of the ERCW model to the OCPC model, a model of parallel computing based on dynamically reconfigurable optical networks, and of BFO circuits to the OCPC model with limited dynamic reconfiguration ability.

1 Introduction

In this paper we develop algorithms and lower bounds for fundamental problems on the Exclusive Read Concurrent Write (ERCW) Parallel Random Access Machine (PRAM) model. The ERCW PRAM model has not received much attention, due in part to a general belief that concurrent writing does not add much power to a model without concurrent reading. We show that this is not always the case by presenting algorithms that solve problems on the ERCW PRAM much faster than they could be solved on the EREW PRAM. (See [21] for more details on the different PRAM models.) We further motivate the ERCW by its relation to parallel computers with optical communication networks. Since there is no 'queue' delay in optical communication networks, the ERCW PRAM is a better model for parallel machines with such networks than the recently proposed QRQW (or ERQW) model [15].

Many results for the ERCW PRAM follow directly from results for the EREW PRAM or CRCW PRAM. For instance, the global OR of n bits can be found in constant time on an n processor ERCW PRAM, as on a CRCW PRAM, but broadcasting 1 bit to n processors requires $\Theta(\log n)$ steps, as on an EREW PRAM. The result for broadcasting implies that computing the prefix sums of n inputs and merging two lists of size n both require $\Theta(\log n)$ time also. However, some results obtained directly from EREW PRAM and CRCW PRAM results do not give tight bounds. For instance, the problem of computing the parity of n bits on the ERCW PRAM has a lower bound of $\Omega(\log n/\log\log n)$ on the CRCW PRAM, and an upper bound of $O(\log n)$ from the EREW PRAM. Tight bounds are not known for the ERCW PRAM. Furthermore, tight bounds

* This research was supported by Texas Advanced Research Projects Grant 003658480. Current address: Boise State University, Boise, ID 83725 (philmac@cs.idbsu.edu)

** This research was supported in part by Texas Advanced Research Projects Grants 003658480 and 003658386, and NSF Grant CCR 90-23059. (vlr@cs.utexas.edu) are not known for many other problems, including the problems of compaction and finding the maximum. In this paper, however, we make significant progress in developing tighter bounds for these and other problems.

Our results for the ERCW PRAM (here n is the size of the input, and all algorithms perform linear work except as noted) include a k-compaction algorithm that runs in $O(\log\log n + \log k)$ time; a randomized algorithm for k-compaction that runs in $O(\log k)$ expected time; a randomized algorithm for approximate k-compaction that runs in $O(\log\log k)$ time, with failure probability $1/k$; an algorithm for finding the maximum of inputs in the range $[1, n]$ that runs in $O(\log\log n)$ time; an algorithm for chaining that runs in $O(\log\log n)$ time; an algorithm for integer chain-sorting (linear-size integers) that runs in $O(\log\log n)$ time; and an algorithm for integer sorting (polynomial-size integers) that runs in $O(\log n)$ using almost linear work.

We present two lower bounds results for the ERCW PRAM: a lower bound of $\Omega(\sqrt{\log\log n})$ time for solving compaction, and a lower bound of $\Omega(\sqrt{\log n})$ for finding the maximum of general inputs. (The former was discovered independently by Goldberg and Jerrum, and led to the $\Omega(\sqrt{\log\log n})$ lower bound on h-relation routing in Goldberg, Jerrum and MacKenzie [17].)

Finally, we consider unbounded fan-in, bounded fan-out (BFO) circuits. The computations on such circuits can be mapped optimally onto an ERCW PRAM as oblivious algorithms. We show that any BFO circuit for adding two n-bit integers, merging a bit into an n bit sorted sequence, sorting n bits, computing the prefix sums or parity of n bits requires $\Omega(\log n)$ depth. Let $TH_{k,n}$ denote the threshold function which outputs 1 if and only if at least k of the inputs are equal to 1. We show that $TH_{k,n}$ can be computed by a linear size, $O(\log\log n + \log k)$ depth circuit, and that any BFO circuit which computes $TH_{k,n}$ requires $\Omega(\log\log n + \log k)$ depth.

As further motivation for studying the ERCW PRAM model we show that it is related to a model of massively parallel computing based on dynamically reconfigurable optical networks. Specifically, we show that the ERCW PRAM with n global memory cells and unlimited local memory is computationally equivalent to the OCPC (Optical Communication Parallel Computer) model [1, 13, 14, 16, 27] on n processors (This is in contrast to the statement given in [1] that the OCPC model is equivalent to an EREW PRAM with n global memory cells.) Since the OCPC model uses full dynamic reconfiguration, which is not yet technically feasible, we are also interested in developing oblivious ERCW PRAM algorithms, which only require partial dynamic reconfiguration. This motivates the study of BFO circuits, which provide these oblivious ERCW PRAM algorithms.

The current interest in the OCPC model, the close relation between the OCPC model and the ERCW PRAM model, and the richness of results obtained so far on the OCPC, the ERCW PRAM, and the BFO circuit model, all indicate that these are important models of parallel computation which should be studied further.

Due to space limitations, we will sketch or omit many of the proofs. Details can be found in [24].

2 Preliminaries

An Exclusive Read, Concurrent Write (ERCW) PRAM consists of a collection of processors, each with infinite local memory, which operate synchronously and communicate through a global memory. Each read or write to global memory takes one time step. Only one processor can read from any memory cell at any time step, but multiple processors may write to a memory cell in a single time step. Write conflicts are handled according to one of the following standard collision resolution protocols: Priority, Arbitrary, Common, Collision, Tolerant and Robust [19]. (Since the standard OCPC model uses the Tolerant protocol, we will be most concerned with developing ERCW PRAM algorithms using the Tolerant protocol. We define the OCPC model in Section 6.)

The ERCW(ack) PRAM is an ERCW PRAM with the added feature that a processor which successfully writes to a cell receives an acknowledgement. To retain the spirit of the Common model, we assume no processor receives an acknowledgement in the Common model. To retain the spirit of the Robust model, we assume that false "successful" writes could cause bogus acknowledgements to be sent.

Often we would like to separate the issues of using the global memory as storage for inputs and outputs, and using the global memory for communication. In these cases, we will assume that inputs and outputs are spread evenly among the local memories of the processors. For instance, given p processors and n inputs, we will assume each processor contains n/p inputs in its local memory. With this assumption, we will be free to design algorithms which use less than n cells of global memory.

In our algorithms we do not require that all processors learn the output of an algorithm, for this would force a trivial $\Omega(\log n)$ time lower bound on all our algorithms.

Lemma 1. *An n processor ERCW(ack) PRAM with m global memory cells can be simulated on a $\max\{n, m\}$ processor ERCW PRAM with $2m+n$ global memory cells with the same write conflict protocol (except Robust).*

Proof Sketch. On the simulating machine, use the second m memory locations to determine the successfully writing processor (if any), and the last n locations for acknowledgements. Details omitted. □

3 Compaction problems

In this section, we study the problems of k-compaction and approximate k-compaction on the ERCW PRAM. The *k-compaction* problem takes an array of size n with k marked elements, and places the marked elements into an array of size k. The *approximate k-compaction* problem takes an array of size n with k marked elements, and places the marked elements into an array of size $O(k)$. Compaction and approximate compaction are important subproblems in processor reallocation and load balancing. First we give a lower bound.

Theorem 2. *Solving 2-compaction on a Robust, Common, Collision, or Tolerant ERCW PRAM requires $\Omega(\sqrt{\log\log n})$ steps, and for $k \le \sqrt{(\log\log n)/2} - 1$, solving k-compaction on a Priority or Arbitrary ERCW PRAM requires at least k steps.* □

Proof Sketch. We sketch the lower bound for 2-compaction on the Tolerant ERCW PRAM. Note that as in [9] we simply need to prove a lower bound on the 2-OR problem (the OR problem where at most 2 of the inputs are '1'). In our case, either zero or exactly two inputs will be '1'.

A step will consist of a write followed by a read. At each step, an adversary will fix some inputs to either '0' or '1'. Live inputs are inputs that have not been fixed. Let V_t be the set of indices of inputs which have not been fixed after step t, where $V_0 = \{1, \ldots, n\}$. Let p_t be the maximum number of processors which could be affected by a given live input. Let c_t be the maximum number of cells which could be affected by a given live input. Let $k_t = \max\{c_t, p_t\}$.

Lemma 3. *We can construct an adversary such that after step t,* (1) *$k_t \le 4^t$;* (2) *$|V_t| \ge |V_{t-1}|^{1/k_{t-1}}/152k_{t-1}^2$; and* (3) *each processor and cell is affected by at most one live input.*

Proof Sketch. We prove this by induction. First note that $p_0 = 0 < 4^0$, $c_0 = 1 = 4^0$, each processor is affected by no inputs, and each cell is affected by at most one input.

Now assume the lemma is true up to step t. Then we show how to make it hold for step $t+1$. Let $k = k_t$. Say a processor P *zero-writes* to a cell C at step $t+1$ if (1) it writes to that cell if the input that affects P is 0, and (2) that input is still live. Define *one-writes* analogously. We omit the proof that the adversary can fix (to 0) all but $m = |V_t|/(2k+1)$ inputs so that each cell either is unchanged (possibly due to a collision), or has at most one processor that zero-writes to it.

Next we show that an adversary can fix (to 0) all but $m^{1/k}/k$ of the inputs such that if a cell is one-written to then either it is the only cell one-written to, or no cell is one-written to by more than one processor. To show this, we find a "sunflower" in a group of sets, where each set contains the cells one-written to by processors which know a given live input. By the Erdös-Rado Theorem [6], there must be a sunflower of size $(m/k!)^{1/k} \ge m^{1/k}/k$. Note that if a cell is the only cell one-written to, then there will be a collision at that cell (because two inputs are 1). Let $m' = m^{1/k}/k \ge |V_t|^{1/k}/2k$.

It is easy to see that at most $c' = c_t + 2p_t$ cells and at most $p' = c' + p_t$ processors are affected by any single live input after the write step. Also, each cell could be affected by at most three live inputs and each processor could be affected by at most four live inputs. Applying Turan's Theorem, we show that the adversary can fix all but $m'/(18p'+1) \ge |V_t|^{1/k}/152k^2$ inputs such that each cell and processor is affected by at most one live input. □

Plugging $T = \sqrt{(\log\log n)/2)} - 1$ into Lemma 3 we find that for large n, $|V_T| \ge 3$. Thus there will be 2 live inputs which do not affect the output cell, but they could affect the result of the OR (e.g., if all the other inputs are fixed to 0). □

We note that there is a simple deterministic algorithm which solves k-compaction in $O(k)$ time on an Arbitrary ERCW PRAM. However, this algorithm will not work unless some processor can succeed in each write. For the other write conflict resolution protocols we need a different approach.

The algorithm and details of the following upper bound can be found in [24].

Theorem 4. *Let $t(n,k) = \log\log n + \log k$. The k-compaction problem can be solved in $O(t(n,k))$ time on an $n/t(n,k)$ processor Collision or Tolerant ERCW PRAM with $n/t(n,k)$ global memory cells, and on an $n/t(n,k)$ processor Robust ERCW PRAM with $O((n/t(n,k))^2)$ global memory cells.*

We now present two results for randomized algorithms for compaction. Both results are obtained by having processors hash into random locations in an array. In the first algorithm for the Robust ERCW PRAM we simply hash elements into an array of size k^4, compress this array using a prefix operation, and test whether all elements have succeeded. In the second algorithm for the Tolerant ERCW(ack) PRAM each processor with a marked element writes it to a random location in an array of size $8k$. If a processor receives an acknowledgement, it idles. If not, the processor writes its element into an array of size $4k$. This procedure continues for a total of $\log\log k$ steps as the array size reduces by half each time. Then we attempt for three steps to write the remaining elements into arrays of size k. This leads to the following two theorems.

Theorem 5. *An $n/\log k$ processor Robust ERCW PRAM with no more than $n/\log k$ global memory cells can solve k-compaction in $O(\log k)$ expected time.*

Theorem 6. *An $n/(\log\log k)$ processor Tolerant ERCW PRAM with $n/(\log\log k)$ global memory cells can solve approximate k-compaction in time $O(\log\log k)$, with probability $1 - 1/k$.*

4 Finding the Maximum, Chaining and Integer Sorting

In this section we summarize our results for three basic problems. Details and proofs of these results can be found in [24].

4.1 Finding the Maximum

We summarize here our results for the problem of finding the maximum of n inputs and for the Global OR problem (i.e., the problem of finding the maximum of n bits). Details can be found in [24].

Finding the maximum of n inputs requires $\Theta(\log n)$ time on an EREW or CREW PRAM, even when the inputs are restricted to be either 0 or 1 [4]. Finding the maximum of n inputs on a Priority CRCW with n processors requires

$\Theta(\log\log n)$ time if the inputs come from a large range and $O(k)$ time if the inputs are restricted to the range $[1, n^k]$ [11]. In contrast we show two lower bounds for the ERCW PRAM, followed by upper bounds that improve on results derived from the other PRAM models.

Theorem 7. *Finding the maximum of n inputs on a Priority ERCW PRAM requires $\Omega(\sqrt{\log n})$ communication steps.*

Theorem 8. *Finding the maximum of n inputs drawn from the range $[0, s]$, for $s < n$, requires $\Omega(\sqrt{\log\log s})$ time on a Robust, Tolerant, Collision, or Common ERCW PRAM.*

Proof. Consider an input array of size n which consists of all zeros except for two entries at locations $i, j \in [1, s]$, which contain the values i and j, respectively. Solving 2-compaction in this array can easily be reduced to finding the maximum of the n inputs, and thus the $\Omega(\sqrt{\log\log s})$ lower bound on 2-compaction applies to the problem of finding the maximum. □

We now state upper bound results for the maximum problem and the global OR problem.

Theorem 9. *The maximum of n inputs in the range $[0, s]$ can be found on a $\max\{n, s\}/\log\log s$ processor Common, Tolerant, or Collision ERCW PRAM in $O(\log\log s)$ time.*

Theorem 10. *An $n/\log\log n$ processor Robust ERCW PRAM can find the global OR of n bits in $\Theta(\log\log n)$ time with error probability $\frac{1}{n}$.*

4.2 Chaining and Integer Sorting

Given an input of n bits, the Chaining problem is the problem of determining for each 1 input, the position of the nearest one to its left. Given an input of n integers, the Integer Chain-sorting problem is the problem of obtaining a linked list of these n integers in sorted in order. The proofs of the following results can be found in [24].

Theorem 11. *The Chaining problem on n bits can be solved on an $n/\log\log n$ processor Common, Tolerant, or Collision ERCW PRAM in $O(\log\log n)$ time.*

Theorem 12. *Integer chain-sorting can be performed on n integers in the range $[0..n-1]$ in $O(\log\log n)$ time with n processors on a Priority ERCW(ack) PRAM.*

Theorem 13. *Integer sort into an array can be performed on n integers in the range $[0..n^k]$ in $O(\log n)$ time with $n\log\log n/\log n$ processors and $n^{1+\epsilon}$ space on a Priority ERCW(ack) PRAM.*

5 Unbounded Fan-in, Bounded Fan-out (BFO) Circuits

We assume standard definitions for circuits and formulas [2]. A BFO circuit with size s and depth d can be simulated in a straightforward way by an s processor, d step oblivious OCPC algorithm. Just as unbounded fan-in, unbounded fan-out circuits correspond closely to the CRCW PRAM [3], and the study of bounded fan-in circuits often sheds light on problems on the CREW and EREW PRAM, we believe that the study of BFO circuits should enhance the understanding of the ERCW PRAMs.

We now present results on solving some fundamental problems on BFO circuits. Our first result shows how to transform a BFO circuit into something resembling a formula, so that we can obtain a lower bound the depth of the circuit using known lower bounds on formula size.

Theorem 14. *Let f be a Boolean function over n variables. If a circuit of depth d with fan-out at most c (with one input corresponding to each variable) computes f, then there is a Boolean formula of size at most nc^d which computes f.*

Proof Sketch. Consider a gate with fan-out $c > 1$. Create an equivalent circuit by duplicating the gate c times, and moving the fan-out to the inputs of the gate. Continue until each gate has fan-out 1. Then each input will have a fan-out of at most c^d, and thus the formula will have size nc^d. □

Corollary 15. *Any BFO circuit which computes parity requires $\Omega(\log n)$ depth.*

Proof. By Khrapchenko [22], any formula for parity must have size $\Omega(n^2)$. By the previous lemma, $nc^d = \Omega(n^2)$, and since c is a constant, $d = \Omega(\log n)$. □

Let $TH_{k,n}$ denote the threshold function which outputs 1 if and only if at least k of the inputs are equal to 1.

Corollary 16. *Any BFO circuit which computes $TH_{k,n}$ requires $\Omega(\log k + \log\log n)$ depth.*

Proof. By Khrapchenko [22] any formula for $TH_{k,n}$ must have size $\Omega(k(n-k+1))$. By Krichevskii [23] any formula for $TH_{k,n}$ must have size $\Omega(n \log n)$. By the previous lemma, $nc^d = \max\{\Omega(k(n-k+1)), \Omega(n \log n)\}$, and since c is a constant, $d = \Omega(\log k + \log\log n)$. □

We next consider the computation of multiple-valued Boolean functions.

Lemma 17. *Let $f : R^n \to R^m$ be a Boolean function. Consider the jth input variable for some $j, 1 \leq j \leq n$. Let O be a set of output variables with the property that for each $o \in O$ there is some n-bit input I such that the value of o is complemented when the jth bit in I is complemented. Then any bounded fan-out circuit that computes f will require depth $\Omega(\log |O|)$.*

Proof. The circuit must contain a path from the jth input node to each of the output nodes in O. Since the circuit has bounded fan-out, the lemma follows. □

Corollary 18. *Any bounded fan-out circuit for adding two n-bit integers, merging a bit into an n bit sorted sequence, sorting n bits, or computing the prefix sums of n bits requires $\Omega(\log n)$ depth.*

There are well known bounded fan-in circuits with $O(\log n)$ depth and linear size for parity, addition, merging, sorting binary inputs, and prefix sums on binary inputs. By [20], these circuits can be converted into bounded fan-out circuits of the same size and depth. By Corollaries 15 and 18, these are optimal.

Next, we show that by building on constructions in Muller and Preparata [25], Valiant [26], and Friedman [12], we can construct a BFO circuit which computes the threshold function $TH_{k,n}$ in optimal size n, and optimal depth $O(\log k + \log\log n)$. We omit the proof.

Theorem 19. *There is a size $O(n)$, depth $O(\log k + \log\log n)$ BFO circuit which computes $TH_{k,n}$.*

6 Optical Communication and ERCW PRAMS

Here we describe the technology for optical communication, the OCPC model which is derived from this technology, and its relation to the ERCW PRAM.

6.1 Optical Communication Technology

There are two basic types of optical interconnection networks, fiber optic networks, and free-space optic networks. The type of fiber optic network which allows unit time communication between any pairs of processors is the *Passive optical star coupler* network [5]. In this network all processors are connected via optical fibers to a passive optical star coupler, which broadcasts messages sent from one processor to all other processors. To allow more flexible communication, time division multiplexing (TDM) or wavelength division multiplexing (WDM) is used. For unit time communication, we must use (WDM). For dynamic reconfiguration ability, we must have tunable transmitters and/or receivers. Currently, tunable transmitters and receivers are too slow to be practical.

6.2 OCPC model

One abstraction of the passive optical star coupler model was first considered by Anderson and Miller [1], and has since been studied in [7, 13, 14, 16, 17, 27]. Various names for this model have been proposed, including *Local Memory PRAM*, S*PRAM, OMC, OCP, and OCPC. We will use the term OCPC, denoting *Optical Communication Parallel Computer.*

An OCPC consists of a collection of processors, each with infinite local memory, which operate synchronously and communicate by transmitting messages to each other. At any step, a processor can transmit at most one message to another processor. The message will succeed in reaching the processor if it is the only message being sent to that processor at that step. Concurrent transmissions to the same processor are handled according to one of the following collision resolution protocols: Priority, Arbitrary, Common, Collision, Tolerant and Robust [19]. (Note that the standard OCPC model uses the Tolerant protocol.)

The OCPC(ack) is an OCPC with the added feature that a processor which successfully transmits a message to another processor receives an acknowledgement as in the ERCW PRAM (see Section 2).

The following are some relationships between OCPC and ERCW PRAM models, with and without acknowledgements. Proofs are omitted due to space limitations.

Lemma 20. *An n processor OCPC can be simulated on an n processor ERCW PRAM with n global memory cells with the same write conflict protocol. Also, an n processor ERCW PRAM (ERCW(ack) PRAM) with m global memory cells can be simulated on a $\max\{n, m\}$ processor OCPC with the same write conflict protocol (except the Robust ERCW(ack) PRAM).*

References

1. R. J. Anderson and G. L. Miller. Optical communication for pointer based algorithms. Technical Report CRI 88-14, University of Southern California, 1988.
2. R. B. Boppana and M. Sipser. The complexity of finite functions. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, chapter 14, pages 757–804. MIT Press/Elsevier, 1990.
3. A. K. Chandra, L. J. Stockmeyer, and U. Vishkin. A complexity theory for unbounded fan-in parallelism. In *Proc. 23th Symp. on Found. of Comp. Sci.*, pages 1–13, 1982.
4. S. Cook, C. Dwork, and R. Reischuk. Upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM J. Comput.*, 15(1):87–97, February 1986.
5. P. Dowd. High performance interprocessor communication through optical wavelength division multiple access channels. In *Proc. 18th Symp. on Comp. Arch.*, pages 96–105, 1991.
6. P. Erdös and R. Rado. Intersection Theorems for Systems of Sets. *J. London Math.*, 35:85–90, 1960.
7. M. M. Eshaghian. Parallel algorithms for image processing on omc. *IEEE Trans. Comput.*, 40(7):827–833, 1991.
8. F. Fich, R. Impagliazzo, B. Kapron, V. King, and M. Kutylowski. Limits on the power of parallel random access machines with weak forms of write conflict resolution. In *Proc. of 10th Symp. on Theor. Aspects of Comp. Sci.*, page unknown, 1993.
9. F. Fich, M. Kowaluk, M. Kutylowski, K. Loryś, and P. Ragde. Retrieval of scattered information by EREW, CREW, and CRCW PRAMs. In *Proc. 3rd Scand. Workshop on Alg. Theory*, pages 30–41. Lec. Notes in Comp. Sci., Vol. 621, 1992.
10. F. E. Fich, F. Meyer auf der Heide, P. Ragde, and A. Wigderson. One, two, three . . . infinity: Lower bounds for parallel computation. In *Proc. 17th Symp. on Theory of Computing*, pages 48–58, 1985.

11. F. E. Fich, P. Ragde, and A. Wigderson. Relations between concurrent-write models of parallel computation. *SIAM J. Comput.*, 17:606–627, 1988.
12. J. Friedman. Construct $O(n \log n)$ size montone formulae for the kth threshold function of n boolean variables. *SIAM J. Comput.*, 15(3):641–654, 1986.
13. A. V. Gerbessiotis and L. G. Valiant. Direct bulk-synchronous parallel algorithms. In *Proc. Scandinavian Workshop on Algo. Theory*, 1992.
14. M. Geréb-Graus and T. Tsantilas. Efficient optical communication in parallel computers. In *Proc. ACM Symp. on Para. Alg. and Arch.*, pages 41–48, 1992.
15. P. B. Gibbons, Y. Matias, V. Ramachandran. The Queue-Read Queue-Write PRAM model: Accounting for contention in parallel algorithms. In *Proc. ACM-SIAM Symp. on Discrete Algs.* 1994, *SIAM J Comput*, to appear.
16. L. A. Goldberg, M. Jerrum, T. Leighton, and S. Rao. A doubly logarithmic communication algorithm for the completely connected optical communication parallel computer. In *Proc. ACM Symp. on Para. Alg. and Arch.*, pages 300–309, 1993.
17. L. A. Goldberg, M. Jerrum, and P. D. MacKenzie. A lower bound for routing on a completely connected optical communication parallel computer. accepted to SPAA, 1994.
18. T. Hagerup. Towards optimal parallel bucket sorting. *Inform. and Comp.*, 75:39–51, 1987.
19. T. Hagerup and T. Radzik. Every robust CRCW PRAM can efficiently simulate a Priority PRAM. In *Proc. 2nd ACM Symp. on Para. Alg. and Arch.*, pages 117–124, 1990.
20. H. J. Hoover, M. M. Klawe, and N. J. Pippenger. Bounding fan-out in logical networks. *J. Assoc. Comput. Mach.*, 31(1):13–18, 1984.
21. R. M. Karp and V. Ramachandran. Parallel algorithms for shared-memory machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, chapter 17, pages 869–941. MIT Press/Elsevier, 1990.
22. V. M. Khrapchenko. A method for determining lower bounds for the complexity of Π-schemes. *Mat. Zametki*, 10(1):83-92, 1971. (in Russian); English translation in: *Math. Notes* **10**(1) (1971) 474-479.
23. R. E. Krichevskii. Complexity of contact circuits realizing a function of logical algebra. *Dokl. Akad. Nauk SSSR*, 151(4):803–806, 1963. (in Russian); English translation in: *Soviet Phys. Dokl.* 8(8) (1964) 770–772.
24. P. D. MacKenzie and V. Ramachandran. ERCW PRAMs and Optical Communication. TR96-16, Dept. of Comp. Sci., Univ of Texas at Austin, 1996.
25. D. E. Muller and F. P. Preparata. Bounds to complexities of networks for sorting and for switching. *J. Assoc. Comput. Mach.*, 22(2):195–201, April 1975.
26. L. G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 5:363–366, 1984.
27. L. G. Valiant. General purpose parallel architectures. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, chapter 18, pages 945–971. MIT Press/Elsevier, 1990.