Parallel Processing Letters

INFORMATION BROADCASTING BY EXCLUSIVE-READ PRAMS

PAUL BEAME*
Dept. of Computer Science and Engineering, FR-35 University of Washington, Seattle, Washington 98195, USA
email: beame@cs.washington.edu

and

MIROSŁAW KUTYŁOWSKI† and MARCIN KIK‡
Institute of Computer Science, University of Wrocław, Przesmyckiego 20, PL-51-151 Wrocław, Poland
email: mirekk@uni-paderborn.de and kik@plwruw11.bitnet

Received (received date)
Revised (revised date)
Communicated by (Name of Editor)

ABSTRACT

We consider the problem of copying information stored initially in a single memory cell by Exclusive Read Exclusive Write Parallel Random Access Machines (EREW PRAMs). We prove lower bounds for this problem and present algorithms matching them tightly (in many cases up to an *additive* constant). The bounds presented depend on the number of cells used and size of information copied. The lower bounds apply also to functions where a change of a single argument influences the output in many memory locations.

Keywords: PRAM; exclusive read

1. Introduction

A parallel random access machine or PRAM (for a formal definition see [4]) consists of a collection of processors and memory cells. Each computation step consists of three phases performed simultaneously by all processors. During the first phase each processor reads a chosen memory cell (or stays idle). During the second phase the processors perform internal computations. During the third phase each processor may write into an arbitrarily chosen memory cell. If no processors attempt to write simultaneously into the same cell, then we call the PRAM a concurrent read exclusive write (CREW) PRAM; if additionally no processors read the same cell at the same

*Supported by NSF grant CCR-8858799 and NSF/DARPA grant CCR-8907960
† Partially supported by Polish Academy of Sciences (program CPBP 01.01), Alexander-von-Humboldt Foundation from Bonn and KBN grant 2 1197 91 01
‡ Partially supported by KBN grant 2 1197 91 01

time we call it an exclusive read exclusive write (EREW) PRAM. If a PRAM has to compute a multi-output function $f : D^n \rightarrow R^m$, $f(\mathbf{x}) = <f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})>$, then initially the input arguments are placed in the first n memory cells, each argument in a separate cell; and after the last computation step the values $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})$ are stored in m fixed memory cells, each value in a separate cell.

The many variants of PRAMs have been extremely useful abstractions for describing parallel algorithms. However, in comparing two algorithms for a problem on different PRAM variants (especially algorithms with different time bounds) it is not clear how to decide which is preferable. A natural question to consider then is the relative power of different PRAM models and this has been studied by many authors. In particular, a great deal of attention has been given to comparing the relative power of the CREW PRAM and many variants of the concurrent read, concurrent write (CRCW) PRAM.

The question of how a prohibition of simultaneous *reads* affects the PRAM model, in particular the relationship between the EREW PRAM and the CREW PRAM, has received somewhat more limited attention despite the fact that the EREW restriction is arguably more practical. (For example, the simulations of PRAMs by networks due to Ranade [10] has additional complications to handle concurrent memory access and these are at least as bad for reading as for writing.) Snir [11] showed a time separation of $\Omega(\sqrt{\log n})$ between CREW and EREW PRAMs for the problem of searching a sorted list and this has been extended to a problem with full domain [7]. A major drawback of these results is that they use multi-variate Ramsey theory and thus require huge input integers. Later, Fich and Wigderson [6] were able to compare exclusive and concurrent reading for Boolean input domain and obtain a similar separation for a decision problem based on decision trees. However their lower bound only holds when a restriction is placed on the kind of writing allowed, the *owner* write assumption, i.e. they separate EROW from CROW PRAMs. (The models are further restricted by having only one global memory cell per processor.)

We address the question of obtaining a separation between the EREW and CREW PRAM models for problems with reasonable, say Boolean, domains. The problem we consider is that of *information broadcast*: given $x \in \{0, \ldots, k-1\}$ in single memory location, copy x so that it is stored in m fixed memory locations (including the input location). This problem is the key to step-by-step simulation of CREW by EREW PRAMs since it captures the essence of the power of concurrent reading – a CREW PRAM can do this in a single step. An EREW PRAM broadcasting algorithm can be easily obtained from the algorithm of Cook, Dwork and Reischuk [3] for computing the OR of n bits. To create m copies, it requires $\phi(m) \approx 0.72 \log m$[§] steps, and uses only m processors and cells for any k (a precise definition of $\phi(m)$ is given below). We present a complete analysis of the broadcasting problem on the EREW PRAM and prove tight time bounds. The complexity of this problem ranges between, roughly speaking, $0.52 \log m$ and $0.72 \log m$ depending on k and the number of processors and cells used. These results are summarized in Theorem 1.

Definition 1. Let $\beta(m) = \lceil \log_{2+\sqrt{3}} m \rceil$. The Fibonacci numbers F_i, $i \geq 0$, are defined by $F_0 = 0$, $F_1 = 1$, and $F_{i+2} = F_{i+1} + F_i$ for $i \geq 0$. Let $a = \frac{1+\sqrt{5}}{2}$. It is known that $a^{t-2} \leq F_t \leq a^{t-1}$, for $t > 0$. Let $\phi(x) = \min\{t : F_{2t+1} \geq x\}$. (Then

[§]Throughout the paper, log stands for log to the base 2.

$\log_b m \le \phi(m) \le \log_b m{+}1.34$, where $b = a^2 = a + 1$, i.e., $\phi(m) \approx 0.72 \log m$.)

Theorem 1. *Let T be the number of steps necessary for a EREW PRAM to broadcast $x \in \{0, \dots, k-1\}$ into m memory cells. Then:*

(i) *for any k, $\beta(m) \le T$, independent of the number of processors and memory cells* (Corollaries 1 and 3),

(ii) *for $k = 2$, $T \le \beta(m){+}1$, even if the algorithm can use only m processors and m cells* (Theorem 3),

(iii) *for arbitrary k, $T \le \beta(m){+}3$, provided that $k \cdot m$ processors and $k \cdot m$ cells may be used* Theorem 4),

(iv) *if only m processors and m cells may be used during a computation, then*

 (a) *for $k \ge (m+1)^{m-1}$, $T = \phi(m)$* ([3], see Theorem 5, and Theorem 7),

 (b) *for $k < (m+1)^{m-1}$, $T \le \min(\phi(m),\ \beta(m) + 1.20 \log\log k + O(1))$* (Theorem 5 and Corollary 4)*, and* $T \ge \beta(m) + 0.19(\log\log k(m+1) - \log\log(m+1)) - 4$ (Corollary 6).

We prove that the lower bound $\lceil \log_{2+\sqrt{3}} m \rceil$ applies to any EREW PRAM that computes a multi-output Boolean function where a change of a single input bit may change m output bits (Theorem 2). This is the only known general lower bound that distinguishes between EREW and CREW PRAMs working on Boolean inputs (for applications of this lower bound see [5, 8]).

This paper represents a combination of results in [1] and [9].

2. EREW PRAMs with unlimited resources

2.1. Lower bounds

Definition 2. (Cook, Dwork and Reischuk [3]). For $x \in \{0,1\}^n$ and $i \in \{1, \dots, n\}$ let x^i denote the unique input y in $\{0,1\}^n$ with $y_j = x_j$, for $j \ne i$, and $y_i \ne x_i$. An input index *i affects a processor P at step t on input $x \in \{0,1\}^n$* if and only if the state of P at step t is different on inputs x and x^i. Similarly say that input index *i affects memory cell C at step t on input x* if and only if the contents of C at step t are different on x and x^i.

Definition 3. Let the sequences $p_0, p_1, \dots$ and $c_0, c_1, \dots$ be defined by

$$p_0 = 0, \quad c_0 = 1; \quad \text{and} \quad p_{i+1} = p_i + c_i, \quad c_{i+1} = 2p_{i+1} + c_i \quad \text{for } i \ge 0.$$

The solution to these recurrences can easily be found by standard techniques:

$$c_t = \tfrac{3+\sqrt{3}}{6}[(2+\sqrt{3})^t + (2-\sqrt{3})^{t+1}], \qquad p_t = \tfrac{\sqrt{3}}{6}[(2+\sqrt{3})^t - (2-\sqrt{3})^t].$$

Since $|2-\sqrt{3}| < 1$, the first term dominates the series and $c_t \approx (2+\sqrt{3})^t$. Also, it follows that for each m, $\min\{t : c_t \ge m\} = \lceil \log_{2+\sqrt{3}} m \rceil \approx 0.526 \log m$.

Lemma 1. *Let M be an EREW PRAM with input $x \in \{0,1\}^n$. Suppose that after step t_0 on input x, an input index $i \in \{1, \dots, n\}$ affects at most r memory cells and no processor of M. Then after step $t_0 + t$, index i affects at most $r \cdot p_t$ processors of M and at most $r \cdot c_t$ memory cells of M.*

Proof. For each $i \in \{1, \dots, n\}$, let $\mathcal{P}_i^t$ be the set of processors that i affects on input x after $t_0 + t$ steps and let $\mathcal{C}_i^t$ be the corresponding set of memory cells. The statement of the theorem is equivalent to $|\mathcal{P}_i^t| \leq r \cdot p_t$ and $|\mathcal{C}_i^t| \leq r \cdot c_t$ for all t. We prove this by induction on t. The base case is trivially true.

Induction Step: Suppose that the bounds are correct after time step t_0+t and consider an input index $i \in \{1, \ldots, n\}$.

Let $P \in \mathcal{P}_i^{t+1} \setminus \mathcal{P}_i^t$. Since i does not affect processor P after t_0+t steps on input x then the state of processor P is the same after t_0+t steps on both x and x^i. Thus there is at most one memory cell C_j that processor P reads on both inputs x and x^i during step t_0+t+1. It follows that since i affects P after t_0+t+1 steps it must have affected C_j after t_0+t steps, i.e. $C_j \in \mathcal{C}_i^t$. Furthermore no other processor $P' \in \mathcal{P}_i^{t+1} \setminus \mathcal{P}_i^t$ can read memory cell C_j on input x or x^i since P already does so and this is an EREW PRAM. Thus we have $|\mathcal{P}_i^{t+1} \setminus \mathcal{P}_i^t| \leq |\mathcal{C}_i^t|$. Using the inductive hypothesis it follows that $|\mathcal{P}_i^{t+1}| \leq r \cdot p_t + r \cdot c_t = r \cdot p_{t+1}$ as required.

Now let $C \in \mathcal{C}_i^{t+1} \setminus \mathcal{C}_i^t$. Since $C \notin \mathcal{C}_i^t$, there must be some processor $P_j \in \mathcal{P}_i^{t+1}$ that writes into memory cell C during step t_0+t+1 on either x or x^i. Since each processor writes into at most one memory cell on a given input at a given time step, there can be at most $2|\mathcal{P}_i^{t+1}|$ memory cells that are written into by some processor in $\mathcal{P}_i^{t+1}$ on either x or x^i during step t_0+t+1. Thus $|\mathcal{C}_i^{t+1} \setminus \mathcal{C}_i^t| \leq 2|\mathcal{P}_i^{t+1}|$ and we have $|\mathcal{C}_i^{t+1}| \leq r \cdot c_t + 2 \cdot r \cdot p_{t+1} = r \cdot c_{t+1}$ as required. $\square$

Definition 4. For $f : \{0,1\}^n \to R$, input index i is *critical for f on input x* if and only if $f(x) \neq f(x^i)$. The *fan-out* of a multi-output function $f : \{0,1\}^n \to R^m$, $f(\mathbf{x}) = <f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})>$ is the maximum over all $x \in \{0,1\}^n$ and $i \in \{1, \ldots, n\}$ of the number of output components $j \in \{1, \ldots, m\}$ such that i is critical for f_j on input x.

Theorem 2. *Any function of Boolean inputs with fan-out m requires at least $\lceil \log_{2+\sqrt{3}} m \rceil$ steps to be computed on an EREW PRAM.*

Proof. Let x be the input and i be the index that achieve the maximum in the definition of the fan-out of f. If the function is computed in T steps, then it must be that each memory cell C_j with an output $f_j(x)$ must be affected by i on input x at step T since input i is critical for f_j on input x. Thus applying Lemma 1 for $t_0 = 0$ and $r = 1$, we get $c_T \geq m$. Hence $T \geq \min\{t : c_t \geq m\} = \lceil \log_{2+\sqrt{3}} m \rceil$. $\square$

Since information broadcast has high fan-out we have

Corollary 1. *Information broadcast of a single Boolean value to m memory cells requires $\lceil \log_{2+\sqrt{3}} m \rceil$ steps on an EREW PRAM.*

Also, since a CREW PRAM with p processors can do an information broadcast to p memory cells in 1 step, we have

Corollary 2. *Any general step-by-step simulation on Boolean inputs of a CREW PRAM with p processors by an EREW PRAM requires $\Omega(\log p)$ time per step.*

Although this is nice confirmation of our intuition, it remains an open problem to show a separation between EREW and CREW PRAMs for decision problems on Boolean inputs as opposed to the multi-output functions described here.

We finally note that Theorem 2 can be useful when the problem in question does not have Boolean inputs. Consider the *m-Expand* problem: given an m bit integer $b_m \cdots b_1$ stored in C_0, produce as output the bits $b_1, \ldots, b_m$ in memory cells $C_1, \ldots, C_m$ respectively. It is easy to see that one could solve m cell information broadcast of a bit using an algorithm for the *m-Expand* problem by interpreting 0 as the integer 0 and 1 as the integer $2^m - 1$.

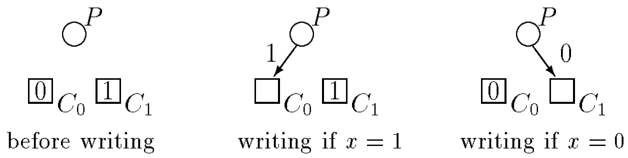

Fig. 1. Sending a bit x into two cells in one step.

Corollary 3. *Computing m-Expand on an EREW PRAM requires* $\lceil \log_{2+\sqrt{3}} m \rceil$ *steps.*

2.2. Time optimal broadcasting algorithms

In this subsection we present broadcasting algorithms that match the lower time bound of Corollary 1.

The first algorithm presented below uses a simple trick: during one writing phase one processor can store a number $x \in \{0, 1\}$ in two different cells, say C_0 and C_1. Indeed, if cells C_0 and C_1 are prepared so that initially C_i stores number i, then in either case we need to change the contents of only one cell. Indeed, it suffices to write x into cell C_{1-x} (see figure 1). (This trick can also be applied to compute the PARITY of n bits on an EREW PRAM in $\approx 0.86 \log n$ steps [4].)

Theorem 3. *Information broadcast of a value* $x \in \{0, 1\}$ *to* m *memory cells can be performed in* $\lceil \log_{2+\sqrt{3}} m \rceil + 1$ *steps by an EREW PRAM with* m *processors and* m *cells.*

Proof. Let x be stored in cell C_0. During step 0, for each $i = 1, 2, \ldots, m-1$, processor P_i writes into cell C_i. The number written is 0 if i is even and 1 if i is odd. Then the main procedure starts. It has the following properties: after step $t-1$,

(i) the processors $P_0, P_1, \ldots, P_{p_{t-1}-1}$ know x,

(ii) the cells $C_0, \ldots, C_{c_{t-1}-1}$ store x; the other cells contain the numbers written to them during step 0.

During step t, for each $i \in \{0, \ldots, c_{t-1}-1\}$, processor $P_{p_{t-1}+i}$ reads cell C_i. Then for each $i \in \{0, \ldots, p_{t-1}-1\} \cup \{p_{t-1}, \ldots, p_{t-1}+c_{t-1}-1\} = \{0, \ldots, p_t-1\}$, processor P_i knows x, as required. During the writing phase of step t, for each $i \in \{0, \ldots, p_t-1\}$ processor P_i writes x into cell $C_{c_{t-1}+2i}$ if $x = 0$ or into cell $C_{c_{t-1}+2i+1}$ if $x = 1$. So processor P_i always "corrects" the cells $C_{c_{t-1}+2i}$ and $C_{c_{t-1}+2i+1}$ so that they both contain x, as described above. We see that after the writing phase the following cells store x: $C_0, \ldots, C_{c_{t-1}-1}$ (the cells storing x before step t) and $C_{c_{t-1}}, \ldots, C_{c_{t-1}+2(p_t-1)+1}$ (the cells "corrected" at step t). But $c_{t-1}+2(p_t-1)+1 = c_t-1$, hence the cells storing x are now $C_0, \ldots, C_{c_t-1}$, as required.

The computation terminates after step T of the main procedure if there are m copies of x, i.e. for $T = \min\{t : c_t \geq m\} = \lceil \log_{2+\sqrt{3}} m \rceil$. $\square$

Theorem 4. *Information broadcast of a value* $x \in \{0, \ldots, k-1\}$ *to* m *memory cells can be performed in* $\lceil \log_{2+\sqrt{3}} m \rceil + 3$ *steps by an EREW PRAM with* $(m-1) \cdot k$ *processors and* $m \cdot k$ *cells.*

Proof. There are k groups of auxiliary memory cells, each consisting of $m-1$ cells. The cells in each group are numbered from 0 through $m-2$. During the first step, the cells are initialized so that the odd numbered cells store 1's and the even numbered cells store 0's. At the next step one processor reads x from the input cell C_0 and writes 1 into cell 0 of group x. Then within each group we perform the algorithm from Theorem 3 to copy the contents of cell 0 into all cells of the group. The initial step of this algorithm (i.e. step 0) has been already performed so it takes $\lceil \log_{2+\sqrt{3}}(m-1)\rceil$ steps. Afterwards all cells except the cells of group x contain 0's and the cells of group x contain 1's. During the last step, $(m-1)\cdot k$ processors read all auxiliary cells in parallel. If a processor encounters 0 then it halts; if it encounters 1, say in cell j of group x, then it writes x into the output cell C_{j+1}. $\Box$

Remark 1. A counting argument shows that almost all Boolean functions of n variables require at least $n-2\log n-9$ steps on a PRIORITY PRAM with n processors and memory cells able to store only single bits [2]. It is interesting to see that the computation time can be reduced even for EREW PRAMs by using exponentially many processors: *Any single output function of n Boolean variables can be computed by an EREW PRAM with cells able to store only bits and $n\cdot 2^n$ processors in $\lceil \log_{2+\sqrt{3}} 2^n\rceil + O(\log n) \leq 0.53n + O(1)$ steps.* Indeed, this can be done as follows: using the algorithm of Theorem 3, 2^n copies of the input are created in $\lceil \log_{2+\sqrt{3}} 2^n\rceil + 1$ steps. Then, separately for each sequence s of n bits, a group of n processors checks whether s coincides with the input string. This takes $O(\log n)$ steps by the algorithm of Cook, Dwork and Reischuk [3]. The processor that detects the coincidence writes the output value.

3. EREW PRAMs with limited resources

3.1. Efficient broadcasting algorithms

In this subsection we present two broadcasting algorithms using only m processors and m cells. The first one, due to Cook, Dwork and Reischuk [3], makes $\phi(m)$ steps to create m copies of $x \in \{0,\dots,k-1\}$. We shall later see that this is optimal for $k \geq (m+1)^{m-1}$. For small values of k, the second algorithm gives a run time of $\beta(m) + 1.20\log\log k + O(1)$ which is close to optimal.

Theorem 5. (Cook, Dwork and Reischuk [3]) *Information broadcast of an arbitrary value x to m cells can be performed in $\phi(m)$ steps by an EREW PRAM with m processors and m cells.*

Proof. (Sketch) At step t, for each $i \in \{0,\dots,F_{2t-1}-1\}$, processor $P_{F_{2t-2}+i}$ reads the value of x from cell C_i and then for each $i \in \{0,\dots,F_{2t}-1\}$, processor P_i writes x to cell $C_{F_{2t-1}+i}$. $\Box$

Lemma 2. *Let the binary representation of a number x be stored in k cells, each bit in a separate cell. There is an EREW PRAM with $2^{\lceil \log k\rceil}$ processors and cells such that after step $\lceil \log k\rceil + 1$ every processor knows x and every cell stores x.*

Proof. By adding leading zeros we may assume that $k = 2^i$, for some i. Then we proceed by induction on i. For $i = 0$ the lemma is obviously true. If $i > 0$, then the halves of the binary representation of x represent two numbers, say x_1 and x_2. We apply the algorithm given by the induction hypothesis to x_1 and to x_2. Then each of the 2^{i-1} processors knowing x_1 (knowing x_2) reads from one of the 2^{i-1} cells storing x_2 (storing x_1). After reading, each processor knows x and stores x in a cell. $\Box$

Theorem 6. *Let $s = \lceil \log k\rceil$. Information broadcast of a value $x \in \{0,\dots,k-1\}$ to m memory cells can be performed in $\beta(m/s) + \phi(s) + \lceil \log s\rceil + 3$ steps by an EREW*

PRAM with m processors and m cells.

Proof. Let $l = \lceil \frac{m}{2s} \rceil$. The algorithm consists of three phases.

Phase 1: x is copied into s cells using the algorithm from Theorem 5. This uses $\phi(s)$ computation steps, s processors and s memory cells.

Phase 2: each bit of the binary representation of x is written into l different memory cells. The processors and cells are divided into s groups. The processors in group i write the ith bit of the binary representation of x into l different cells using the ith cell storing x as its input cell. The algorithm from Theorem 3 is used; it requires $\beta(l)+1 = \beta(\lceil \frac{m}{2s} \rceil)+1 \leq \beta(m/s)+1$ computation steps and $s \cdot l \leq m$ processors and cells.

Phase 3: the processors work independently in l groups; each group writes x into s different memory cells. Namely, each group uses a group of cells storing the binary representation of x and performs the algorithm from Lemma 2 to create s copies of x. It takes $\lceil \log s \rceil + 1$ steps; in total, $l \cdot 2^{\lceil \log s \rceil} < 2ls \leq m$ processors and cells are used. The number of the copies of x created is equal to $l \cdot s = \lceil \frac{m}{2s} \rceil \cdot s \geq \frac{m}{2}$. In a single additional step we can increase the number of copies to m.

Clearly, the algorithm runs in the stated time. □

Corollary 4. *Information broadcast of a value $x \in \{0, \ldots, k-1\}$ to m memory cells can be performed in $\beta(m) + 1.20 \log\log k + O(1)$ steps by an EREW PRAM with m processors and m cells.*

Proof. Let $s = \lceil \log k \rceil$ and let d be such that $\phi(n) - \beta(n) = d \log n \pm O(1)$. Then $.19 < d < .20$. By Theorem 6, the time for this broadcast is at most

$$\begin{aligned} \beta(m/s) + \phi(s) + \lceil \log s \rceil + 3 &= \beta(m) - \beta(s) + \phi(s) + \log s + O(1) \\ &= \beta(m) + (1+d) \log\log k + O(1) \\ &\leq \beta(m) + 1.20 \log\log k + O(1). \quad \square \end{aligned}$$

Corollary 4 improves Theorem 5 (except for an additive constant) if $\phi(m) > \beta(m) + (1+d)\log\log k + O(1)$, and this is true if $k \leq 2^{m^{d/(1+d)}}$ which holds if $k \leq 2^{m^{.16}}$.

3.2. Lower bounds

Lemma 3. *Let M be an EREW PRAM with only m memory cells $C_0, \ldots, C_{m-1}$ and input value $x \in \{0, \ldots, k-1\}$ given in a single cell C_0. For each step t, there is a subset $X_t \subseteq \{0, \ldots, k-1\}$, such that $|X_t| \geq k/(m+1)^{F_{2t+1}-1}$ and after step t there are at most F_{2t} processors of M whose state is not constant for all inputs in X_t and at most F_{2t+1} memory cells of M whose contents are not constant for all inputs in X_t.*

Proof. We define sets X_t, $\mathcal{P}_t$, $\mathcal{C}_t$ such that after step t a processor not in $\mathcal{P}_t$ (a memory cell not in $\mathcal{C}_t$) is in the same state for each input in X_t. We put $\mathcal{P}_0 = \emptyset$, $\mathcal{C}_0 = \{C_0\}$ and $X_0 = \{0, \ldots, k-1\}$. For $t > 0$, we define

$$\mathcal{P}_t = \mathcal{P}_{t-1} \cup \{P \notin \mathcal{P}_{t-1} : \exists_{C \in \mathcal{C}_{t-1}} \exists_{x \in X_{t-1}}\ P \text{ reads from } C \text{ at step } t \text{ on input } x\}.$$

At the beginning of step t, a processor $P \notin \mathcal{P}_{t-1}$ is in the same state for every input in X_{t-1}. Hence, at step t, if P reads from a certain cell for some input in X_{t-1}, it reads from that that cell for all inputs in X_{t-1}. So by the exclusive read assumption, $|\mathcal{P}_t| \leq |\mathcal{P}_{t-1}| + |\mathcal{C}_{t-1}|$.

For a given set of processors $\mathcal{P}$ and $x \in \{0, \ldots, k-1\}$, let

$$\mathcal{W}_t(x) = \{C : \exists_{P \in \mathcal{P}_t} \; P \text{ writes into cell } C \text{ at step } t \text{ on input } x\}.$$

There are no more than $(m+1)^{|\mathcal{P}_t|}$ possible sets $\mathcal{W}_t(x)$. So we may choose a subset $X_t \subseteq X_{t-1}$ such that $|X_t| \geq |X_{t-1}|/(m+1)^{|\mathcal{P}_t|}$ and $\mathcal{W}_t(x)$ is the same for $x \in X_t$. We define $\mathcal{C}_t = \mathcal{C}_{t-1} \cup \mathcal{W}_t(x)$ where $x \in X_t$ is arbitrary. Since every processor writes to at most one memory cell we have $|\mathcal{C}_t| \leq |\mathcal{P}_t| + |\mathcal{C}_{t-1}|$. Combining this with the inequality $|\mathcal{P}_t| \leq |\mathcal{P}_{t-1}| + |\mathcal{C}_{t-1}|$, we get that $|\mathcal{P}_t| \leq F_{2t}$ and $|\mathcal{C}_t| \leq F_{2t+1}$. It remains to estimate $|X_t|$. Since $|X_j| \geq |X_{j-1}|/(m+1)^{|\mathcal{P}_j|}$ for each j, it follows that $|X_t| \geq k/(m+1)^s$ where $s = \sum_{i=1}^{t} F_{2i}$. Now $\sum_{i=1}^{t} F_{2i} = F_{2t+1} - F_1 = F_{2t+1} - 1$ so the result follows immediately. □

Corollary 5. *Let T be the time required to broadcast $x \in \{0, \ldots, k-1\}$ to m memory cells by an EREW PRAM with only m memory cells. Then* $T \geq \min(\phi(m), \phi(1 + \log_{m+1} k))$.

Proof. Suppose $T < \phi(1 + \log_{m+1} k)$. Then by definition of ϕ, $F_{2T+1} < 1 + \log_{m+1} k$ and thus $k > (m+1)^{F_{2T+1}-1}$. Rewriting, we get $k/(m+1)^{F_{2T+1}-1} > 1$, so by Lemma 3 we can choose a set $X_T \subseteq \{0, \ldots, k-1\}$ such that $|X_T| > 1$ and there are at most F_{2T+1} memory locations whose contents immediately after step T is not constant for all inputs in X_T. Since $|X_T| > 1$, any memory location receiving the broadcast cannot be constant, so at most F_{2T+1} memory locations may store the input value. Thus $T \geq \phi(m)$. □

We now directly apply Corollary 5 for large values of k but for small values of k we combine it with the lower bound for $k = 2$ to get better results.

Theorem 7. *Let T be the time required to broadcast $x \in \{0, \ldots, k-1\}$ to m memory cells by an EREW PRAM with only m memory cells. Then*

(i) $T \geq \phi(m)$, *if* $k \geq (m+1)^{m-1}$,

(ii) $T \geq \beta(m/s') + \phi(s') - 2$ *for* $s' = 1 + \log_{m+1} k$, *if* $k < (m+1)^{m-1}$.

Proof. If $k \geq (m+1)^{m-1}$ then $1 + \log_{m+1} k \geq m$ so using Corollary 5 and the fact that ϕ is nondecreasing we obtain $T \geq \phi(m)$.

Now assume that $k < (m+1)^{m-1}$ and let $s' = 1 + \log_{m+1} k$. Then $T' = \phi(s') = \phi(1 + \log_{m+1} k) \leq \phi(m)$ so in this case the $\phi(m)$ term plays no role in the minimum calculated in the statement of Corollary 5. By definition of ϕ, $F_{2T'+1} \geq s' > F_{2T'-1}$. Let $X_{T'-1}$ be the set given by Lemma 3. We fix two values $x_1, x_2 \in X_{T'-1}$ and assume that $x \in \{x_1, x_2\}$. Even if this assumption is known to each processor and memory cell, there will be at most $F_{2T'-2}$ processors and $F_{2T'-1}$ cells whose state indicates the value of x. After reading at step T' the number of processors knowing x is at most $F_{2T'-2} + F_{2T'-1} = F_{2T'}$. Hence after this read phase there would have been strictly more information broadcast if there had been $F_{2T'}$ memory cells and 0 processors knowing value x just prior to step T'. But even then, according to Lemma 1, M would still require at least $T'' = \beta(m/F_{2T'})$ steps. Now, since $F_{2T'-1} < s'$, $F_{2T'} = F_{2T'-1} + F_{2T'-2} \leq 2F_{2T'-1} < 2s'$. Therefore $T \geq T' - 1 + T'' \geq \phi(s') + \beta(m/2s') - 1 \geq \phi(s') + \beta(m/s') - 2$ by the definition of β. □

Corollary 6. *Let T be the time required to broadcast $x \in \{0, \ldots, k-1\}$ to m memory cells by an EREW PRAM with only m memory cells. Then if $k < (m+1)^{m-1}$, $T \geq \beta(m) - .19(\log\log k(m+1) - \log\log(m+1)) - 4$.*

Proof: Let $s' = 1 + \log_{m+1} k = \frac{\log k(m+1)}{\log(m+1)}$ and suppose that $k < (m+1)^{m-1}$. Applying Theorem 7(ii), T is at least

$$\begin{aligned} \beta(m/s') + \phi(s') - 2 &= \beta(m) - \beta(s') + \phi(s') - 3 \\ &\geq \beta(m) + (\log_b 2 - \log_{2+\sqrt{3}} 2) \log s' - 4 \\ &\geq \beta(m) + .19 \log(\frac{\log k(m+1)}{\log(m+1)}) - 4 \\ &\geq \beta(m) + .19(\log\log k(m+1) - \log\log(m+1)) - 4. \quad \square \end{aligned}$$

Acknowledgements

The authors thank the anonymous referees for their criticism and very detailed remarks that helped to improve quality of the paper. The third author thanks Martin Dietzfelbinger for many discussions on the subject of the paper.

References

1. P. Beame, Lower bounds in parallel machine computation, Ph.D. Thesis, University of Toronto, Technical Report TR198a/87, 1987.
2. S.J. Bellantoni, Parallel RAMs with bounded memory wordsize, *Inform. and Comput.* **91** (1991) 259–273.
3. 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)** (1986) 87–97.
4. M. Dietzfelbinger, M. Kutyłowski and R. Reischuk, Exact time bounds for computing Boolean functions on PRAMs without simultaneous writes, in *Proc. 2nd ACM Sympos. Parallel Algorithms and Architectures*, 1990, 125–135.
5. F. Fich, M. Kowaluk, M. Kutyłowski, K. Loryś and P. Ragde, Retrieval of scattered information by EREW, CREW and CRCW PRAMs, in *Proc. 3rd Scandinavian Workshop on Algorithm Theory*, 1992, 30-41. (LNCS 621, Springer Verlag, 1992).
6. F. Fich and A. Wigderson, Towards understanding exclusive read, in *Proc. 1st ACM Sympos. Parallel Algorithms and Architectures*, 1989, 76-82; to appear in *SIAM J. Comput.*.
7. A. Gafni, J. Naor and P. Ragde, On separating the EREW and CROW models, to appear in *Theoretical Computer Science*.
8. T. Hagerup and M. Kutyłowski, Fast integer merging on the EREW PRAM, in *Proc. of International Coll. on Automata Languages and Programming*, 1992, 318–329. (LCNS 623, Springer Verlag, 1992).
9. M. Kik and M. Kutyłowski, Information broadcasting by Exclusive Read PRAMs, *manuscript*, 1990.
10. A. Ranade, How to emulate shared memory, in *Proc. 28th IEEE Sympos. Found. Computer Science*, 1987, 185–194.
11. M. Snir, On parallel searching, *SIAM J. Comput.* **14(4)** (1985) 688–708.