Impossibility Results for Weak Threshold Networks ⋆

Costas Busch [a,1] and Marios Mavronicolas [b,2]

[a] *Department of Computer Science, Brown University, Providence, RI 02912*

[b] *Department of Computer Science, University of Cyprus, Nicosia 1678, Cyprus*

Abstract

It is shown that a *weak threshold network* (in particular, *threshold network*) of width w and depth d cannot be constructed from balancers of width $p_0, p_1, \ldots, p_{m-1}$, if w does not divide P^d, where P is the least common multiple of $p_0, p_1, \ldots, p_{m-1}$. This holds regardless of the size of the network, as long as it is finite, and it implies a lower bound of $\log_P w$ on its depth. More strongly, a lower bound of $\log_{p_{\max}} w$ is shown on the length of *every* path from an input wire to any output wire that exhibits the threshold property, where $p_{\max}$ is the maximum among $p_0, p_1, \ldots, p_{m-1}$.

Keywords: Distributed computing, parallel processing, impossibility results.

1 Introduction

Consider a distributed application which involves solving a system of equations by successive relaxation, where each process holds part of the data. Interleaving of steps by different processes is necessary in order to ensure that a correct

⋆ Some of the results in this paper were announced in preliminary form in the conference version of [6].

[1] Supported by NSF research grants DMS 95-05949 and CCR 96-13785. Part of the work of this author was done while at Department of Computer Science, University of Crete, and Institute of Computer Science, Foundation for Research and Technology – Hellas.

[2] Partially supported by ESPRIT III Basic Research Project # 8144 – LYDIA ("Load Balancing on High Performance Parallel and Distributed Systems"), and by funds for the promotion of research at University of Cyprus (research projects "Load Balancing Problems on Shared-Memory Multiprocessor Architectures" and "Distributed, Parallel, and Concurrent Computations"). Part of the work of this author was done while visiting Institute of Computer Science, Foundation for Research and Technology – Hellas.

value was computed, since it implies sufficient interaction among the intermediate values computed by the processes. A simple way to solve this problem makes use of a *synchronization barrier,* a distributed data structure ensuring that no process advances beyond a particular point in a computation until all processes have arrived at that point (see, e.g., [3,8,10]). Conventional multiprocessors mostly use centralized barriers, which often become the network bottleneck or "hot-spots" in the shared memory.

In a seminal paper, Aspnes *et al.* [2] suggest a completely different approach to such synchronization problems in the context of *balancing networks,* a new class of distributed data structures suitable for solving more general *balancing problems.* Their idea is to use a collection of primitive computing elements called *balancers*, each having low expected contention, in a way that a process needs to access only a few elements in order to advance to the next computation phase. Informally, a p-balancer can be thought of as a p-input, p-output toggle. When an input appears on one of the input wires, it takes the output wire to which the toggle is set, and toggles the gate, so that the next input will leave on the next output wire.[‡] We say that a p-balancer has *width* p.

One can "connect" a collection of balancers to form a *balancing network* much in the same way a comparator network is obtained by connecting a collection of comparators (see, e.g., [12]). This is done by connecting output wires from some balancers to input wires of other balancers. The remaining unconnected input and output wires are the input and output wires, respectively, of the network. Each access to a balancing network by a process corresponds to a *token* which traverses the network from an input wire to an output wire. Let x_i (resp., y_j) denote the number of tokens that have entered (resp., left) the network on the ith input wire (resp., jth output wire), where $0 \leq i, j \leq w-1$ for some integer $w \geq 2$ called *width.* A balancing network is a *threshold network* if each time the network becomes free of tokens, $y_{j_0} = \lfloor \sum_{l=0}^{w-1} x_l/w \rfloor$, where j_0 is some fixed but arbitrary output wire. Say that j_0 *exhibits the threshold property.* A *weak threshold network* relaxes the requirement that j_0 be fixed.

Aspnes *et al.* [2, Section 5.3] describe an implementation of a barrier for n processes, where $n \equiv 0 \bmod w$, using a threshold network of width w; their implementation is an adaptation of the "sense-reversing" technique of Hensgen *et al.* [10]. Substituting a weak threshold network for a threshold network in that implementation still guarantees correctness, although it may not allow optimizations that were possible when using threshold networks (e.g., associating a local counter with only one output wire). Thus, it would be necessary to construct weak threshold networks of arbitrary width in order to adapt to arbitrary dynamic changes in the number of processes ac-

[‡] A balancer can be implemented both on a shared-memory multi-processor machine and in message-passing [2,14].

cessing a barrier implemented that way. This paper addresses the problem of constructing weak threshold networks of arbitrary width. More precisely, what is the width of weak threshold networks that can be constructed using a finite (but unbounded) number of balancers whose width is in the set $\{p_0, p_1, \ldots, p_{m-1}\}$? An additional motivation for our study is the existence of general tradeoffs between depth, width and contention in balancing networks (see, e.g., [5,7,9,11]). [§] An optimal weak threshold network for a fixed number of processes may have to optimize both width and contention; the optimal width may be arbitrary.

We show that if w does not divide P^d, then there is no weak threshold (in particular, threshold) network of width w and depth d, where P is the least common multiple of $p_0, p_1, \ldots, p_{m-1}$. Moreover, this impossibility result immediately implies a lower bound of $\log_P w$ on the maximal path length, *viz.* depth, of any weak threshold network. However, our combinatorial proof techniques yield a strictly stronger impossibility result. We show that *every* path from an input wire to any output wire that exhibits the threshold property must have length at least $\log_{p_{\max}} w$, where $p_{\max}$ is the maximum among $p_0, p_1, \ldots, p_{m-1}$.

The problem of constructing networks of arbitrary width using balancers whose width is in a fixed set has been studied before by Aharonson and Attiya [1], Busch and Mavronicolas [6] and Moran and Taubenfeld [13] for *counting* and *smoothing networks,* and generalizations of them. Busch and Mavronicolas [6] show corresponding lower bounds on distances for counting and smoothing networks. Our results generalize all of these results and strictly strengthen them, since any smoothing network is also a weak threshold network but not vice versa; moreover, our proof techniques are completely different from all other published techniques of similar results in being purely combinatorial. Brit *et al.* [4] present impossibility results and lower bounds for *public counters,* satisfying several kinds of correctness conditions for concurrent counting; these results are similar in flavor to ours.

The rest of this paper is organized as follows. In Section 2, we outline a combinatorial framework for the study of balancing networks. Weak threshold, threshold and smoothing networks are introduced and studied in Section 3. Section 4 derives impossibility results for weak threshold networks. We conclude, in Section 5, with a discussion of our results.

[§] The depth of a balancing network is the length of the longest path from an input wire to an output wire.

2 Framework

In this Section, we present a combinatorial framework for the study of balancing networks. Our presentation closely follows the one in [6], where the reader is referred for a more detailed treatment.

For any integer $w \geq 2$, $\mathbf{X}^{(w)}$ denotes the vector $\langle x_0, x_1, \ldots, x_{w-1} \rangle^{\mathrm{T}}$, while $\lceil \mathbf{X}^{(w)} \rceil$ and $\lfloor \mathbf{X}^{(w)} \rfloor$ denote the integer vectors $\langle \lceil x_0 \rceil, \lceil x_1 \rceil, \ldots, \lceil x_{w-1} \rceil \rangle^{\mathrm{T}}$ and $\langle \lfloor x_0 \rfloor, \lfloor x_1 \rfloor, \ldots, \lfloor x_{w-1} \rfloor \rangle^{\mathrm{T}}$, respectively. We use $\mathbf{0}^{(w)}$ to denote $\langle 0, 0, \ldots, 0 \rangle^{\mathrm{T}}$, a vector with w zero entries. For any integer $p \geq 1$, denote $[p] = \{0, 1, \ldots, p-1\}$. In all of our discussion, we will refer to a set $\mathcal{P} = \{p_0, p_1, \ldots, p_{m-1}\}$ of positive integers no less than two, and we will let $p_{\max}$ and P denote the maximum and the least common multiple, respectively, of integers in $\mathcal{P}$.

Fix any integer $p \geq 2$, and let $\sum_{i \geq 0} x_i p^i$ denote the representation of the integer $x \geq 0$ in the p-ary arithmetic system, where, for each i, $x_i \in [p]$. For any integer $k \geq 1$, define $x \downarrow_p k = \sum_{0 \leq i \leq k-1} x_i p^i$ and $x \uparrow_p k = \sum_{i \geq k} x_i p^i$; that is, $x \downarrow_p k$ is the integer represented by the k least significant p-ary digits in the representation of x in the p-ary arithmetic system, while $x \uparrow_p k$ is the integer obtained from this representation by setting each of those digits to zero. Clearly, $x \downarrow_p k + x \uparrow_p k = x$. Furthermore, define $x \updownarrow_p k = x_{k-1} p^{k-1}$; that is, $x \updownarrow_p k$ is the integer represented by the kth least significant p-ary digit of x. The definitions of $x \downarrow_p k$, $x \uparrow_p k$ and $x \updownarrow_p k$ involving the integer x can be extended component-wise to any vector $\mathbf{X}^{(w)}$ in the natural way.

Balancing networks are constructed from computing elements called balancers and wires. For each integer $p \geq 2$, a *p-balancer* $b_p : \mathbf{X}^{(p)} \rightarrow \mathbf{Y}^{(p)}$, or *balancer* for short, is a computing element which receives non-negative, integer inputs $x_0, x_1, \ldots, x_{p-1}$ on input wires $0, 1, \ldots, p-1$, respectively, and computes non-negative integer outputs $y_0, y_1, \ldots, y_{p-1}$ on output wires $0, 1, \ldots, p-1$, respectively, such that for each j, $0 \leq j \leq p-1$, $y_j = \lceil (\sum_{l=0}^{p-1} x_l - j)/p \rceil$. We say that a p-balancer has *width* or *type* p. For each j, $0 \leq j \leq p-1$, the *order* of output wire j, denoted $\mathrm{ord}(j)$, is defined to be j/p. A *balancer over* $\mathcal{P}$ is a p-balancer for some $p \in \mathcal{P}$.

A *balancing network* $\mathcal{B} : \mathbf{X}^{(w)} \rightarrow \mathbf{Y}^{(w)}$ *of width* w *over* $\mathcal{P}$ is a collection of balancers over $\mathcal{P}$, where output wires are connected to input wires, having w designated input wires $0, 1, \ldots, w-1$ (which are not connected to output wires of balancers), w designated output wires $0, 1, \ldots, w-1$ (similarly not connected to input wires of balancers), and containing no cycles. Non-negative, integer inputs $x_0, x_1, \ldots, x_{w-1}$ are received on input wires $0, 1, \ldots, w-1$, respectively, and non-negative, integer outputs $y_0, y_1, \ldots, y_{w-1}$ are computed on output wires $0, 1, \ldots, w-1$, respectively, in the natural way. For a balancing network $\mathcal{B}$, the *depth* of $\mathcal{B}$, denoted $\mathit{depth}(\mathcal{B})$, is defined to be the maximal

depth of any of its wires, where the depth of a wire is defined to be zero for an input wire of $\mathcal{B}$, and $\max_{l \in [p]} depth(x_l) + 1$, for an output wire of a p-balancer with input wires $x_0, x_1, \ldots, x_{p-1}$.

In case $depth(\mathcal{B}) = 1$, $\mathcal{B}$ will be called a *layer*. For a layer $\mathcal{B}$, we define a matrix $\mathbf{D}_{\mathcal{B}}$ with w rows and w columns, called the *distance matrix,* which determines the distance between input and output wires. Formally, for any i and j, $0 \leq i, j \leq w-1$, $\mathbf{D}_{\mathcal{B}}[ji] = 1$ if input wire i and output wire j are connected via a balancer, else $\mathbf{D}_{\mathcal{B}}[ji] = 0$ if input wire i and output wire j coincide, and ∞ otherwise.

If $depth(\mathcal{B}) = d$ is greater than one, then $\mathcal{B}$ can be uniquely partitioned into layers $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_d$ from left to right in the obvious way. We inductively extend the definition of the distance matrix from layers to arbitrary balancing networks as follows. Let $\mathcal{B}'$ be the balancing network resulting from $\mathcal{B}$ by removing its rightmost layer $\mathcal{B}_d$. Then, $\mathbf{D}_{\mathcal{B}}[ji] = \min_{l \in [p]}(\mathbf{D}_{\mathcal{B}'}[jl] + \mathbf{D}_{\mathcal{B}_d}[li])$, for any i and j, $0 \leq i, j \leq w-1$. That is, the distance of input wire i and output wire j is the minimum over all l of the sum of the distance of input wire i and output wire l in $\mathcal{B}'$ and the distance of input wire l and output wire j in $\mathcal{B}'$.

The following result shows that for any balancing network, the output vector takes a particular algebraic form as a function of the input vector, depending on the types of balancers used, and the depth and topology of the network.

Proposition 1 (Busch and Mavronicolas [6]) *Let $\mathcal{B} : \mathbf{X}^{(w)} \rightarrow \mathbf{Y}^{(w)}$ be a balancing network of depth d over $\mathcal{P}$. Then,*

$$\mathbf{Y}^{(w)} \;=\; \mathbf{I}_{\mathcal{B}} \cdot \mathbf{X}^{(w)} \uparrow_P d + \mathbf{F}_{\mathcal{B}}(\mathbf{X}^{(w)} \downarrow_P d)\,,$$

for some matrix $\mathbf{I}_{\mathcal{B}}$ and a vector function $\mathbf{F}_{\mathcal{B}} : [P^d]^w \rightarrow \mathbf{N}^w$.

Call $\mathbf{I}_{\mathcal{B}}$ and $\mathbf{F}_{\mathcal{B}}$ the *transfer parameters* of $\mathcal{B}$. We next sample properties of the transfer parameters that will be used later.

Proposition 2 (Busch and Mavronicolas [6]) *For any balancing network $\mathcal{B} : \mathbf{X}^{(w)} \rightarrow \mathbf{Y}^{(w)}$,*

(1) *the matrix $\mathbf{I}_{\mathcal{B}}$ is doubly stochastic;*
(2) $\mathbf{F}_{\mathcal{B}}(\mathbf{0}^{(w)}) = \mathbf{0}^{(w)}$*;*
(3) *for any i and j, $0 \leq i, j \leq w-1$, $\mathbf{I}_{\mathcal{B}}[ji] \geq 1/p_{\max}^{\mathbf{D}_{\mathcal{B}}[ji]}$.*

3 Weak Threshold, Threshold and Smoothing Networks

In this section, we introduce weak threshold, threshold, and smoothing networks, and present several preliminary properties of them.

Smoothing networks require that outputs come as close to each other as possible.

Definition 3 (Aspnes *et al.* [2]) *A* smoothing network over $\mathcal{P}$ *is a balancing network* $\mathcal{B} : \mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ *over* $\mathcal{P}$ *such that for any* j *and* k, $0 \leq j, k \leq w-1$, $|y_j - y_k| \leq 1$.

A straightforward implication of Definition 3 follows.

Proposition 4 *Assume* $\mathcal{B} : \mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ *is a smoothing network over* $\mathcal{P}$. *Then, for each input vector* $\mathbf{X}^{(w)}$, *there exists at least one* $j \in [w]$ *such that* $y_j = \lfloor \sum_{l=0}^{w-1} x_l/w \rfloor$, *and for each* $j \in [w]$, *either* $y_j = \lfloor \sum_{l=0}^{w-1} x_l/w \rfloor$ *or* $y_j = \lfloor \sum_{l=0}^{w-1} x_l/w \rfloor + 1$.

Our next definition extends the original one of threshold networks over $\{2\}$ given by Aspnes *et al.* [2, Section 5.3].

Definition 5 *A* threshold network over $\mathcal{P}$ *is a balancing network* $\mathcal{B} : \mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ *over* $\mathcal{P}$ *for which there exists some fixed index* $j_0 \in [w]$, *such that for each input vector* $\mathbf{X}^{(w)}$, $y_{j_0} = \lfloor \sum_{l=0}^{w-1} x_l / w \rfloor$.

Roughly speaking, a threshold network can detect input "chunks" of size w on some fixed output wire j_0, called *threshold wire*. Relaxing the requirement that a threshold wire be fixed leads to a weaker definition.

Definition 6 *A* weak threshold network over $\mathcal{P}$ *is a balancing network* $\mathcal{B} : \mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ *over* $\mathcal{P}$ *for which for each input vector* $\mathbf{X}^{(w)}$, *there exists some index* $j_0 = j_0(\mathbf{X}^{(w)}) \in [w]$ *such that* $y_{j_0} = \lfloor \sum_{l=0}^{w-1} x_l / w \rfloor$.

Roughly speaking, a weak threshold network can still detect input "chunks" of size w on some thrshold wire which, however, need not be the same for all inputs. Clearly, any threshold network is also a weak threshold network. Moreover, Proposition 4 immediately implies that weak threshold networks generalize smoothing networks too.

Proposition 7 *Assume* $\mathcal{B}$ *is a smoothing network over* $\mathcal{P}$. *Then,* $\mathcal{B}$ *is a weak threshold network over* $\mathcal{P}$.

Consider a network $\mathcal{B} : \mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ which is the "cascade" of the "parallel composition" of two counting networks, followed by a layer consisting of $w/2$ balancers $b_0, b_1, \ldots, b_{w/2-1}$; for each $i \in [w/2]$, balancer b_i receives as inputs

the ith output of the "top" counting network, and the $(w/2-1-i)$th output of the "bottom" counting network, and produces the corresponding outputs. A straightforward case analysis reveals that $\mathcal{B}$ is a smoothing network. Assume, however, that $x_{w-1} = 0$ while $x_l = 1$ for $l \neq w-1$; by construction of $\mathcal{B}$, it follows that $y_{w-1} = 0$ while $y_l = 1$ for $l \neq w-1$. Thus, output wire $w-1$ is a "candidate" for a threshold wire. Assume now that $x_{w/2-1} = 0$ while $x_l = 1$ for $l \neq w/2-1$; by construction of $\mathcal{B}$, it follows that $y_{w/2-1} = 0$ while $y_l = 1$ for $l \neq w/2-1$. Thus, output wire $w/2-1$ is a second "candidate" for a threshold wire. Since there are two different input vectors for which there correspond different threshold wires, it follows that $\mathcal{B}$ is not a threshold network. Thus, we obtain:

Proposition 8 *Assume $\mathcal{B}$ is a smoothing network over $\mathcal{P}$. Then, $\mathcal{B}$ is not necessarily a threshold network over $\mathcal{P}$.*

Since any threshold network is also a weak threshold network, Proposition 8 immediately implies:

Proposition 9 *Assume $\mathcal{B}$ is a weak threshold network over $\mathcal{P}$. Then, $\mathcal{B}$ is not necessarily a threshold network over $\mathcal{P}$.*

4 Impossibility Results

We present our main impossibility result for weak threshold networks.

Theorem 10 *Assume $\mathcal{B} : \mathbf{X}^{(w)} \rightarrow \mathbf{Y}^{(w)}$ is a weak threshold network of depth d over $\mathcal{P}$. Then, w divides P^d.*

Proof. Fix any index $i \in [w]$, and set $x_i = P^d$ and $x_l = 0$ for $l \neq i$, so that $x_i \uparrow_P d = P^d$, $x_l \uparrow_P d = 0$ for $l \neq i$, $\mathbf{X}^{(w)} \downarrow_P d = \mathbf{0}^{(w)}$ and $\sum_{l=0}^{w-1} x_l \uparrow_P d = P^d$. By Definition 6, there exists some index $j_0 \in [w]$ such that $y_{j_0} = \lfloor P^d/w \rfloor$, while by Proposition 1,

$$y_{j_0} = \sum_{l=0}^{w-1} \mathbf{I}_{\mathcal{B}}[j_0 l]\, x_l \uparrow_P d + \mathbf{F}_{\mathcal{B}}(\mathbf{X}^{(w)} \downarrow_P d)[j_0] = \mathbf{I}_{\mathcal{B}}[j_0 i]\, P^d + \mathbf{F}_{\mathcal{B}}(\mathbf{0}^{(w)})[j_0]\,.$$

By Proposition 2(2), $\mathbf{F}_{\mathcal{B}}(\mathbf{0}^{(w)}) = \mathbf{0}^{(w)}$. It follows that $\lfloor P^d/w \rfloor = \mathbf{I}_{\mathcal{B}}[j_0 i] P^d$, or $\mathbf{I}_{\mathcal{B}}[j_0 i] = \lfloor P^d/w \rfloor / P^d$. Thus, $\mathbf{I}_{\mathcal{B}}[j_0 i]$ is independent of i; since i was chosen arbitrarily, this implies that $\sum_{l=0}^{w-1} \mathbf{I}_{\mathcal{B}}[j_0 l] = w\, \mathbf{I}_{\mathcal{B}}[j_0 i]$. However, by Proposition 2(1), $\sum_{l=0}^{w-1} \mathbf{I}_{\mathcal{B}}[j_0 l] = 1$. Hence, $\mathbf{I}_{\mathcal{B}}[j_0 i] = 1/w$. It follows that $\lfloor P^d/w \rfloor = P^d/w$. This implies that w divides P^d, as needed.

Since any threshold network is a weak threshold network, Theorem 10 immediately implies:

Corollary 11 *Assume $\mathcal{B} : \mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ is a threshold network of depth d over $\mathcal{P}$. Then, w divides P^d.*

An immediate implication of Theorem 10 and Corollary 11 is a lower bound on depth for weak threshold and threshold networks.

Corollary 12 *Assume $\mathcal{B} : \mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ is a weak threshold or threshold network of depth d over $\mathcal{P}$. Then, $d \geq \log_P w$.*

An inspection of the proof of Theorem 10 reveals that for a threshold wire $j_0 \in [w]$ of any weak threshold network $\mathcal{B} : \mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$, $\mathbf{I}_{\mathcal{B}}[j_0 i] = 1/w$ for all $i \in [w]$. Hence, Proposition 2(3) immediately implies a result strictly stronger than Corollary 12.

Theorem 13 *Assume that j_0 is a threshold wire of any weak threshold or threshold network $\mathcal{B} : \mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ over $\mathcal{P}$. Then, for each $i \in [w]$, $\mathbf{D}_{\mathcal{B}}[j_0 i] \geq \log_{p_0} w$.*

5 Discussion

We have derived impossibility results for weak threshold (in particular, threshold) networks. More specifically, we have shown that a weak threshold network of width w and depth d over $\mathcal{P}$ does not exist if w does not divide P^d, and that the length of *every* path from any input wire to an output wire exhibiting the threshold property is at least $\log_{p_{\max}} w$.

The most intriguing question left open by our work is whether a weak threshold (or threshold) network of width w and depth d over $\mathcal{P}$ can be constructed if w divides P^d. Recall that a threshold network is a weak threshold network but not vice versa; on the other hand, a counting network is a threshold network but not vice versa (cf. [2]). Thus, this is a restricted version of a more general question for counting networks originally posed by Aharonson and Attiya [1].

Acknowledgments:

We are thankful to one of the referees for constructive criticisms and significant suggestions.

References

[1] E. Aharonson and H. Attiya, "Counting Networks with Arbitrary Fan-Out," *Distributed Computing,* Vol. 8, pp. 163–169, 1995.

[2] J. Aspnes, M. Herlihy and N. Shavit, "Counting Networks," *Journal of the ACM,* Vol. 41, No. 5, pp. 1020–1048, September 1994.

[3] C. J. Beckmann and C. D. Polychronopoulos, "Fast Barrier Synchronization Hardware," *Proceedings of the IEEE Conference on Supercomputing,* pp. 180–189, 1991.

[4] H. Brit, S. Moran and G. Taubenfeld, "Public Data Structures: Counters as a Special Case," *Proceedings of the 3rd Israel Symposium on the Theory of Computing and Systems,* pp. 98–110, January 1995.

[5] C. Busch, N. Hardavellas and M. Mavronicolas, "Contention in Counting Networks," *Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing,* pp. 404, August 1994.

[6] C. Busch and M. Mavronicolas, "A Combinatorial Treatment of Balancing Networks," *Journal of the ACM,* Vol. 43, No. 5, pp. 794–839, September 1996.

Preliminary version: *Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing,* pp. 206–215, August 1994.

[7] E. W. Felten, A. LaMarca and R. Ladner, "Building Counting Networks from Larger Balancers," Technical Report 93-04-09, Department of Computer Science and Engineering, University of Washington, April 1993.

[8] R. Gupta and C. R. Hill, "A Scalable Implementation of Barrier Synchronization Using an Adaptive Tree," *International Journal of Parallel Programming,* Vol. 18, No. 3, pp. 161–180, June 1989.

[9] N. Hardavellas, D. Karakos and M. Mavronicolas, "Notes on Sorting and Counting Networks," *Proceedings of the 7th International Workshop on Distributed Algorithms (WDAG-93),* Lecture Notes in Computer Science, Vol. # 725 (A. Schiper, ed.), Springer-Verlag, pp. 234–248, Lausanne, Switzerland, September 1993.

[10] D. Hensgen, R. Finkel and U. Manber, "Two Algorithms for Barrier Synchronization," *International Journal of Parallel Programming,* Vol. 17, No. 1, pp. 1–17, 1988.

[11] M. Herlihy, B.-C. Lim and N. Shavit, "Scalable Concurrent Counting," *ACM Transactions on Computer Systems,* Vol. 13, No. 4, pp. 343-364, 1995.

[12] D. Knuth, *The Art of Computer Programming,* Volume 3 (*Sorting and Searching*), Addison-Wesley, 1973.

[13] S. Moran and G. Taubenfeld, "A Lower Bound on Wait-Free Counting," *Proceedings of the 12th Annual ACM Symposium on Principles of Distributed Computing,* pp. 251–259, August 1993.

[14] N. Shavit and A. Zemach, "Diffracting Trees," *ACM Transactions on Computer Systems*, Vol. 14, No. 4, pp. 385-428, 1996.