NORTH-HOLLAND

Informatics and Computer Science

Balancing Networks: State of the Art

MARIOS MAVRONICOLAS

Department of Computer Science, University of Cyprus, Nicosia 1678, Cyprus

ABSTRACT

Balancing networks have recently been proposed by Aspnes et al. (*Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, May 1991, pp. 348–358 as a new class of distributed, low-contention data structures suitable for solving a variety of multiprocessor coordination problems that can be expressed as balancing problems. A significant amount of recent research in multiprocessor computing has been devoted to balancing networks. By way of sampling: constructions of balancing networks satisfying special-purpose properties have been presented and complemented by corresponding inconstructibility results; combinatorial properties of balancing networks have been uncovered, revealing a rich, underlying mathematical structure; the actual performance of balancing networks has been evaluated by both theoretical and experimental means under a variety of degrees of processor concurrency. In this work, I attempt to survey, exemplify, and unify recent research on balancing networks. *© Elsevier Science Inc. 1997*

1. INTRODUCTION

Consider a situation where we have p producers and c consumers. The producers produce jobs at some arbitrary rate; the jobs should be performed by the consumers. One would like to distribute the jobs as evenly as possible among the consumers. A very simple way to solve this problem makes use of a *counter*. When a new job is produced, the producer accesses the counter, increases it, and places the new job in a given array according to the value obtained from the counter. Consumer number l periodically checks the array locations $l, c+l, 2c+l, \ldots$, and whenever any of them contains a new job, the consumer performs it. Clearly, the difference in the total number of jobs eventually performed by any two consumers is at most 1.

INFORMATION SCIENCES **97**, 125–157 (1997)

655 Avenue of the Americas, New York, NY 10010

0020-0255/97/$17.00
PII S0020-0255(96)00175-2

A counter can be easily implemented using a single shared *Fetch& Increment* variable. However, empirically, the time to access a shared variable grows at least linearly with the *contention*, the extent to which concurrent processors simultaneously access the variable.[1] In a seminal paper, Aspnes et al. [6] suggest a completely different approach to such counting problems. Their idea is to use a collection of shared variables, called *balancers*, each having low expected contention, in a way that a processor needs to access only a few variables in order to obtain a value from the counter. Roughly speaking, a balancer can be thought of as a two-output toggle with either one or two inputs. When an input appears on an input wire, it takes the output wire to which the toggle is set, and toggles the gate so that the next input will leave on the other output wire. If the balancer is initialized so that the first input to pass through will exit on the top output wire, then, after m inputs have passed through the toggle, exactly $\lceil m/2 \rceil$ will exit on the top output wire, and $\lfloor m/2 \rfloor$ will exit on the bottom output wire. On a shared-memory multiprocessor machine, a balancer can be implemented by a single-bit *Compare&Swap* variable, and a wire can be implemented by a memory address pointer.

One can "connect" a collection of balancers to form a *balancing network* much in the same way a sorting network is obtained by connecting a collection of comparators (see, e.g., [27]). This is done by connecting output wires from some balancers to input wires of others. The remaining unconnected input and output wires are the input and output wires, respectively, of the network. Each request for a value corresponds to a traversal of the network by a *token*, starting from some input wire, following the pointer obtained by accessing the first balancer to the next one, and so on. Let x_i denote the number of tokens that have entered the network on the ith input wire, $0 \leqslant i \leqslant t-1$, where t is the *input width* of the network. Similarly, y_j denotes the number of tokens that have left the network on the jth output wire, $0 \leqslant j \leqslant w-1$, where w is the *output width* of the network. A balancing network of input width t and output width w is a *counting network* if each time the network becomes free of tokens, i.e., all entering tokens have exited, $0 \leqslant y_i - y_j \leqslant 1$, for any i, j, $0 \leq i < j \leq w-1$. Slightly less demanding, a balancing network of input width t and output width w is a *K-smoothing network* if each time the network becomes free

[1] The cost of contention varies according to the architecture of the system and the specific arbitration protocols used (cf. [5]).

of tokens, i.e., all entering tokens have exited, $0 \leqslant y_i - y_j \leqslant 1$, for any i, j, $0 \leqslant i < j \leqslant w-1$.

Counting and K-smoothing networks are the most well-studied classes of balancing networks. Counting networks have been proven suitable for implementing *shared counters* and *producer/consumer buffers* for multiprocessor architectures [6, 22], while K-smoothing networks are appropriate as hardware solutions to load balancing problems [6, 22, 30]. A large amount of research on balancing networks has attempted to provide constructions of counting and K-smoothing networks and analyze the actual performance of such networks by both theoretical and experimental means [2, 3, 6, 10, 13, 15, 18, 19, 21–23, 25, 26, 32].

One important performance measure for a balancing network is its *depth*, the length of the longest path from an input wire to an output wire, since the depth of the network is equal to the number of memory locations that a processor may have to access before its incremental request has been fulfilled. An important objective of recent research on balancing networks has been to minimize the depth of counting and K-smoothing networks, as a function of the output width w. While a reduction to sorting reveals that depth $\Omega(\log w)$ is necessary [6, 11], there have been plenty of constructions that achieve depth $\Theta(\log^2 w)$ [6, 10, 15, 19, 21, 23]; corresponding experimental evidence suggests that these constructions, although not optimal in depth, perform reasonably well in practice. Not so many constructions come close to the $\Theta(\log w)$ lower bound on depth. The first explicit, deterministic construction of a counting network with input and output width w is one presented by Klugerman and Plaxton [26], achieving depth $\Theta(c^{\log^* w} \log w)$ for some constant c. However, this construction is truly impractical and out of the question to use for practical purposes since it employs the impractical AKS network [4], and this makes the hidden coefficients forbiddenly large.

In all known previous constructions of counting networks [2, 6, 10, 19, 21, 26], the output width w is equal to the input width t, and the constructed counting networks are also sorting networks [6]. Since the only known-to-date sorting network with depth $\Theta(\log w)$ [4] is highly impractical due to large hidden coefficients, insisting that t and w be equal reduces the problem of constructing a "practical" counting network with depth $\Theta(\log w)$ to the problem of constructing a "practical" sorting network with depth $\Theta(\log w)$, remaining open for about twenty-five years [8]. However, insisting that t and w be equal is not really a consequence of the specification of a counting network with w output wires, namely, that it "counts" modulo w. Busch and Mavronicolas use this observation to

provide the first simple, deterministic construction of a counting network with depth $\Theta(\log w)$ [13].

Depth is not the only parameter affecting the actual performance of balancing networks. The time for a processor to traverse a balancing network may also be affected by the simultaneous effort of other processors to traverse the network, and the extent to which a given balancing network allows for such subtle interactions between concurrent processors also deserves study. Recent research has studied the effect of *contention*, the extent to which concurrent processors concurrently access a balancing network, by both experimental and theoretical means. Carefully designed experiments have been conducted to account for the actual performance of balancing networks [6, 10, 19, 21–24, 32]; in a more formal setting, formal complexity models for contention in shared-memory algorithms have been proposed [3, 18] and used to formally explain the reported experimental results (see, e.g., [3, 10, 13, 15, 18, 21]). These formal studies of the performance of balancing networks study the contention as a function of the number n of concurrent processors and the network input and output widths t and w, respectively.

A different research direction attempts to understand the combinatorial structure of balancing networks in a hope that such an improved understanding would reveal any possible fundamental limitations on the performance of balancing networks, which the practitioners should also know, or help design and verify more efficient special-purpose balancing networks. Busch and Mavronicolas [11] develop an elegant, mathematical theory of the combinatorial structure of balancing networks. This theory proves useful for formally showing constructibility and performance (as measured by depth and size) limitations, developing precise algorithms for mathematically verifying that a balancing network meets its specifications, and developing a paradigmatic methodology for showing correctness of general constructions of balancing networks [12, 14].

I apologize for placing perhaps undue emphasis on results I have been involved in, but these are the ones I know best!

The rest of this paper is organized as follows. In Section 2, we present definitions and preliminary facts about balancing networks. In Section 3, we survey constructions of counting networks known so far and their properties. We continue, in Section 4, with an overview of the combinatorial theory of balancing networks presented by Busch and Mavronicolas [11]. Section 5 includes negative results on balancing networks (inconstructibility results and lower bounds). Section 6 surveys experimental and theoretical research on performance of balancing networks. We conclude, in Section 7, with a discussion of the results on balancing networks obtained so far and some directions for further research.

2. DEFINITIONS AND PRELIMINARIES

In this section, we provide a brief introduction to balancing networks; a more complete exposition of the properties of balancing networks can be found in [6, 11].

2.1. BALANCERS AND BALANCING NETWORKS

The construction of balancing networks is similar to that of comparator networks from wires and comparators (see, e.g., [16, Chapter 28] or [27, Section 5.3.4]). We begin by describing balancers.

A *balancer* is a computing element with a number of input wires and a number of output wires. Tokens may arrive on one of the input wires $0, 1, \ldots, r-1$ of the balancer at arbitrary times. Intuitively, a balancer is a toggle mechanism, which, given a stream of input tokens, sends them to output wires $0, 1, \ldots, p-1$ in that order and in a cyclic way; thus, a balancer effectively balances the number of tokens that have been input on its output wires.

The following definitions are tailored for a two-input, two-output balancer. Denote by x_i, $0 \leqslant i \leqslant 1$, the number of input tokens ever received on the balancer's ith input wire, and, similarly, by y_j, $0 \leqslant j \leqslant 1$, the number of tokens ever sent out on its jth output wire. (Throughout, we will sometimes abuse notation and use x_i (resp., y_j) both as the name of the ith input (resp., jth output) wire and the count of the number of input (resp., output) tokens received (resp., sent out) on the wire.) y_0 and y_1 correspond to the *top* and *bottom* output wires, respectively.

The *state* of a two-input, two-output balancer at a given time is defined as the sets of tokens on its input and output wires. For the sake of clarity, we will assume that tokens are all distinct. A state of a two-input, two-output balancer is *quiescent* if $\sum_{i=0}^{1} x_i = \sum_{j=0}^{1} y_j$; that is, the number of tokens that entered the balancer on its two input wires is equal to the number of tokens that left it on its two output wires. The following formal safety, liveness, and step properties are required for a two-input, two-output balancer:

1. In any state, $\sum_{i=0}^{1} x_i \geqslant \sum_{j=0}^{1} y_j$ (i.e., a balancer never creates output tokens).
2. Given any finite number of input tokens $m = \sum_{i=0}^{1} x_i$ to the balancer, the balancer reaches within a finite amount of time a quiescent state (i.e., a balancer never "swallows" input tokens).
3. In any quiescent state, $0 \leqslant y_0 - y_1 \leqslant 1$ (i.e., the output sequence has the *step* property).

Corresponding definitions for a balancer with general numbers of input and output wires follow immediately.

A *balancing network* $\mathscr{B}$ *of input width* t *and output width* w is a collection of balancers, where output wires are connected to input wires, having t designated input wires $0,1,\ldots,t-1$ (with no output wires connected to them), w designated output wires $0,1,\ldots,w-1$ (not connected to input wires), and containing no cycles. A number of tokens $x_0, x_1, \ldots, x_{t-1}$ arrive on input wires $0,1,\ldots,t-1$, respectively, and a number of tokens $y_0, y_1, \ldots, y_{w-1}$ are sent out on output wires $0,1,\ldots,w-1$, respectively. Denote such a network by $\mathscr{B}$: $\mathbf{X}^{(t)} \to \mathbf{Y}^{(w)}$, where $\mathbf{X}^{(t)} = \langle x_0, x_1, \ldots, x_{t-1} \rangle^{\mathrm{T}}$, and $\mathbf{Y}^{(w)} = \langle y_0, y_1, \ldots, y_{w-1} \rangle^{\mathrm{T}}$.

We depict a balancing network of width w as a collection of w horizontal lines with balancers stretched vertically. Figure 1 shows a balancing network, with outputs computed on output wires of all balancers for a specific input. Note that a line does not represent a single wire, but rather a sequence of distinct wires connecting various balancers.

The *size* of a balancing network $\mathscr{B}$, *size*($\mathscr{B}$), is the total number of balancers in the network. The *depth of a wire* y, *depth*(y), is defined to be zero if y is an input wire of the network, and $\max_{i \in [r]} \mathit{depth}(x_i) + 1$, if y is an output wire of a balancer with $r \geqslant 1$ input wires $x_0, x_1, \ldots, x_{r-1}$ and *depth*(x) + 1, if y is an output wire of a one-input balancer with input wire x. The *depth of a balancing network* $\mathscr{B}$, *depth*($\mathscr{B}$), is the maximal depth of any wire.

The *state of a balancing network* is defined as the collection of states of all its component balancers. A state of a balancing network is quiescent if $\sum_{i=0}^{t-1} x_i = \sum_{j=0}^{w-1} y_j$; that is, the number of tokens that entered the network

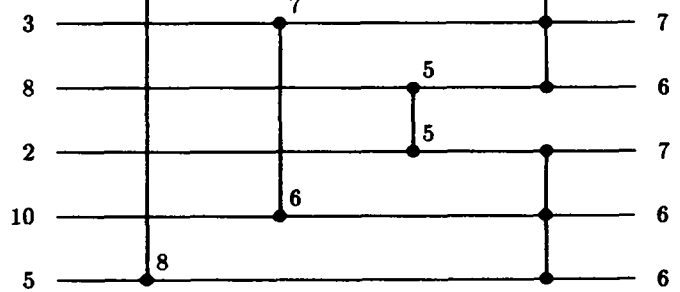

Fig. 1. A balancing network.

is equal to the number of tokens that left it. The safety and liveness properties of a balancing network follow naturally from its definition and the safety and liveness properties of a balancer:

1. In any state, $\sum_{i=0}^{t-1} x_i \geqslant \sum_{j=0}^{w-1} y_j$ (i.e., a balancing network never creates output tokens).
2. Given any finite number of input tokens $m = \sum_{i=0}^{t-1} x_i$ to the network, the network reaches within a finite amount of time a quiescent state (i.e., a network never "swallows" input tokens).

2.2. *CLASSES OF BALANCING NETWORKS*

The most prominent classes of balancing networks studied so far are the counting, K-smoothing, and merging networks.

DEFINITION 2.1 ([6]). A counting network of *fan-in* t and *fan-out* w is a balancing network of these widths for which, in any quiescent state, $0 \leqslant y_j - y_k \leqslant 1$, for any j, k, $0 \leqslant j < k \leqslant w-1$.

That is, the output of a counting network has the step property.

DEFINITION 2.2 ([2]). A *K-smoothing network of fan-in t and fan-out w* is a balancing network of these widths for which, in any quiescent state, $|y_j - y_k| \leqslant K$, for any j, k, $0 \leqslant j, k \leqslant w-1$.

That is, the output of a K-smoothing network has the K-smoothing property. Notice that for any $K \geqslant 1$, a counting network is also a K-smoothing network. A *smoothing network* is a K-smoothing network for any integer $K \geqslant 1$. An alternative way of relaxing Definition 2.1 is to require the step property for the output to hold only if the input has some kind of a step property.

DEFINITION 2.3 ([6]). A *merging network of fan-in t and fan-out w* is a balancing network of these widths for which, in any quiescent state, if $0 \leqslant x_i - x_k \leqslant 1$ for any i, k, $0 \leqslant i < k \leqslant t/2 - 1$, and $0 \leqslant x_i - x_k \leqslant 1$ for any i, k, $t/2 \leqslant i < k \leqslant t-1$, then $0 \leqslant y_j - y_k \leqslant 1$, for any j, k, $0 \leqslant j < k \leqslant w-1$.

That is, the inputs are partitioned into two "blocks," each of size $t/2$, and the output has the step property whenever each of the two input subsequences, one for each of the "blocks," does. Note the resemblance of Definition 2.3 to that of the "classical" comparison-merging networks [27].

3. CONSTRUCTIONS

In this section, we briefly survey some of the main constructions of counting and K-smoothing networks known to date.

Fix throughout w to be any power of 2. We first establish some notation. Let $\mathbf{X}_{up}^{(w/2)}$ and $\mathbf{X}_{down}^{(w/2)}$ denote the vectors $\langle x_0, x_1, \ldots, x_{w/2-1}\rangle^{\mathrm{T}}$ and $\langle xw_{w/2}, x_{w/2+1}, \ldots, x_{w-1}\rangle^{\mathrm{T}}$, respectively. Let also $\mathbf{X}_e^{(w/2)}$ and $\mathbf{X}_o^{(w/2)}$ denote the vectors $\langle x_0, x_2, \ldots, x_{w-2}\rangle$ and $\langle x_1, x_3, \ldots, x_{w-1}\rangle$, respectively. That is, the vectors $\mathbf{X}_e^{(w/2)}$ and $\mathbf{X}_o^{(w/2)}$ contain the odd and even, respectively, entries of $\mathbf{X}^{(w)}$. Finally, let $\mathbf{X}_{eo}^{(w/2)}$ and $\mathbf{X}_{oe}^{(w/2)}$ denote the vectors

$$\langle x_0, x_2, \ldots, x_{w/2-2}, x_{w/2+1}, x_{w/2+3}, \ldots, x_{w-1}\rangle$$

and

$$\langle x_1, x_3, \ldots, x_{w/2-1}, x_{w/2}, x_{w/2+2}, \ldots, x_{w-2}\rangle,$$

respectively. That is, the vector $\mathbf{X}_{eo}^{(w/2)}$ represents the concatenation of the vector of even entries of $\mathbf{X}_{up}^{(w/2)}$ with the vector of odd entries of $\mathbf{X}_{down}^{(w/2)}$, while the vector $\mathbf{X}_{oe}^{(w/2)}$ represents the concatenation of the vector of odd entries of $\mathbf{X}_{up}^{(w/2)}$ with the vector of even entries of $\mathbf{X}_{down}^{(w/2)}$.

Corresponding to the definitions for $\mathbf{X}_{up}^{(w/2)}$, $\mathbf{X}_{down}^{(w/2)}$, $\mathbf{X}_e^{(w/2)}$, $\mathbf{X}_o^{(w/2)}$, $\mathbf{X}_{eo}^{(w/2)}$, and $\mathbf{X}_{oe}^{(w/2)}$, we define index sets $up[w] = \{0, 1, \ldots, w/2-1\}$, $down[w] = \{w/2, w/2+1, \ldots, w-1\}$, $e[w] = \{0, 2, \ldots, w-2\}$, $o[w] = \{1, 3, \ldots, w-1\}$, $eo[w] = \{0, 2, \ldots, w/2-2, w/2+1, w/2+3, \ldots, w-1\}$, and $oe[w] = \{1, 3, \ldots, w/2-1, w/2, w/2+2, \ldots, w-2\}$.

3.1. *THE BITONIC COUNTING NETWORK WITH WIDTH* 2^k

We describe an inductive construction of the bitonic counting network $\mathscr{B}^{(w)}$, following [6]; it uses the bitonic merger network $\mathscr{BM}^{(w)}$, whose construction is described next, as a basic module.

The bitonic merger network $\mathscr{BM}^{(w)}$. The balancing network $\mathscr{BM}^{(w)}$: $\mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$, called *bitonic merger*, is defined inductively as follows: For the base case, where $w = 2$, $\mathscr{BM}^{(2)}$ consists of a single balancer. Assume inductively that we have constructed $\mathscr{BM}^{(w/2)}$, where $w \geqslant 4$; we show how to construct $\mathscr{BM}^{(w)}$. The network $\mathscr{BM}^{(w)}$ is the cascade of:

- a network $\mathscr{N}^{(w)}$: $\mathbf{X}^{(w)} \to \mathbf{Z}^{(w)}$, which is the "parallel composition" of two networks $\mathscr{BM}_{eo}^{(w/2)}$: $\mathbf{X}_{eo}^{(w/2)} \to \mathbf{Z}_e^{(w/2)}$ and $\mathscr{BM}_{oe}^{(w/2)}$: $\mathbf{X}_{oe}^{(w/2)} \to \mathbf{Z}_o^{(w/2)}$;

- a layer $\mathscr{L}^{(w)}$: $\mathbf{Z}^{(w)} \to \mathbf{Y}^{(w)}$ consisting of $w/2$ balancers $b_0, b_1, \ldots, b_{w/2-1}$, where balancer b_i receives inputs z_{2i} and z_{2i+1} and produces outputs y_{2i} and y_{2i+1}, $i \in [w/2]$.

The bitonic network $\mathscr{B}^{(w)}$. The balancing network $\mathscr{B}^{(w)}$: $\mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$, called *bitonic*, is defined inductively. For the base case, where $w = 2$, $\mathscr{B}^{(2)}$ consists of a single balancer. Assume inductively that we have constructed $\mathscr{B}^{(w/2)}$, where $w \geqslant 4$; we show how to construct $\mathscr{B}^{(w)}$. The network $\mathscr{B}^{(w)}$ is the cascade of:

- a network $\mathscr{R}^{(w)}$: $\mathbf{X}^{(w)} \to \mathbf{Z}^{(w)}$, which is the "parallel composition" of two networks $\mathscr{B}_{up}^{(w/2)}$: $\mathbf{X}_{up}^{(w/2)} \to \mathbf{Z}_{up}^{(w/2)}$ and $\mathscr{B}_{down}^{(w/2)}$: $\mathbf{X}_{down}^{(w/2)} \to \mathbf{Z}_{down}^{(w/2)}$;
- a bitonic merger network $\mathscr{BM}^{(w)}$: $\mathbf{Z}^{(w)} \to \mathbf{Y}^{(w)}$.

The bitonic network was one of the two counting networks discovered first [6]. A straightforward proof by induction shows that the bitonic merger network preserves the step property on its inputs shows:

THEOREM 3.1 ([6]). *The network* $\mathscr{BM}^{(w)}$ *is a merging network.*

By construction of $\mathscr{B}^{(w)}$, it immediately follows:

COROLLARY 3.2 ([6]). *The network* $\mathscr{B}^{(w)}$ *is a counting network.*

An alternative proof of Theorem 3.2 has been carried out in [12]. This proof is a concrete instance of a paradigmatic methodology for proving correctness of balancing networks developed in [12] which is, in turn, a direct application of the combinatorial theory of balancing networks proposed in [11] (see Section 4). The new proof provides an interesting complement to the one of Aspnes et al. [6] in terms of modularity and simplicity.

It is trivial to establish that $depth(\mathscr{B}^{(w)}) \in \Theta(\log^2 w)$ and $size(\mathscr{B}^{(w)}) \in \Theta(w \log^2 w)$.

3.2. *A BITONIC COUNTING NETWORK WITH WIDTH* $p2^k$

We describe an inductive construction of the bitonic counting network $\mathscr{B}_p$: $\mathbf{X}^{(p2^k)} \to \mathbf{Y}^{(p2^k)}$, following Busch et al. [10], for any integers $p \geqslant 2$ and $k \geqslant 0$. The construction uses the bitonic merger network $\mathscr{BM}_p^{(p2^k)}$, whose construction is described next, as a basic module.

The bitonic merger network $\mathscr{BM}_p^{(p2^k)}$. The balancing network $\mathscr{BM}_p^{(p2^k)}$, also called *bitonic merger*, is defined inductively as follows:

For the base case, where $k = 1$, the network $\mathscr{BM}_p^{(2p)}$ consists of two layers $\mathscr{L}_2^{(2p)}$ and $\mathscr{L}_p^{(2p)}$; these layers consist solely of 2-balancers and

p-balancers, respectively:

- The layer $\mathscr{L}_2^{(2p)}$: $\mathbf{X}^{(2p)} \to \mathbf{Z}^{(2p)}$ consists of p 2-balancers $b_0^{(2)}, b_1^{(2)}, \ldots, b_{p-1}^{(2)}$. (For each $i \in [p]$, the two input wires of $b_i^{(2)}$ are the ith output wire of $\mathscr{B}_{p,up}^{(2p)}$ and the $(p-1-i)$th output wire of $\mathscr{B}_{p,down}^{(2p)}$.)
- The layer $\mathscr{L}_p^{(2p)}$: $\mathbf{Z}^{(2p)} \to \mathbf{Y}^{(2p)}$ consists of two p-balancers $b_0^{(p)}$ and $b_1^{(p)}$. The p input wires of $b_0^{(p)}$ and $b_1^{(p)}$ are the upper and lower, respectively, output wires of $b_0^{(2)}, b_1^{(2)}, \ldots, b_{p-1}^{(2)}$. The ith output wire of $\mathscr{BM}_p^{(2p)}$ is the ith output wire of $b_0^{(p)}$ for $i \in up[2p]$, and the ith output wire of $\mathscr{BM}_p^{(2p)}$ is the $(i-p)$th output wire of $b_1^{(p)}$ for $p \in down[p]$.

Assume inductively that we have constructed $\mathscr{BM}_p^{(p2^{k-1})}$, where $k \geqslant 2$; we show how to construct $\mathscr{BM}_p^{(p2^k)}$. The network $\mathscr{BM}_p^{(p2^k)}$ is the cascade of:

- a network $\mathscr{N}_p^{(p2^k)}$: $\mathbf{X}^{(p2^k)} \to \mathbf{Z}^{(p2^k)}$, which is the "parallel composition" of two networks $\mathscr{BM}_{p,eo}^{(p2^{k-1})}$: $\mathbf{X}_{eo}^{(p2^k)} \to \mathbf{Z}_{eo}^{(p2^k)}$ and $\mathscr{BM}_{p,oe}^{(p2^{k-1})}$: $\mathbf{X}_{oe}^{(p2^{k-1})} \to \mathbf{Z}_{oe}^{(p2^{k-1})}$;
- a layer $\mathscr{L}_p^{(p2^k)}$: $\mathbf{Z}^{(p2^k)} \to \mathbf{Y}^{(p2^k)}$ consisting of $p2^{k-1}$ 2-balancers $b_0^{(2)}, b_1^{(2)}, \ldots, b_{p2^{k-1}-1}^{(2)}$, where balancer b_i receives inputs z_{2i} and z_{2i+1} and produces outputs y_{2i} and y_{2i+1}, $i \in [p2^k]$.

The bitonic network $\mathscr{B}_p^{(p2^k)}$. The balancing network $\mathscr{B}_p^{(p2^k)}$: $\mathbf{X}^{(p2^k)} \to \mathbf{Y}^{(p2^k)}$, also called *bitonic*, is defined inductively.

For the base case, where $k=0$, $\mathscr{B}_p^{(p)}$ consists of a single p-balancer.

Assume inductively that we have constructed $\mathscr{B}_p^{(2p^{k-1})}$, where $k \geqslant 1$; we show how to construct $\mathscr{B}_p^{(2p^k)}$. The network $\mathscr{B}_p^{(2p^k)}$ is the cascade of:

- a network $\mathscr{R}_p^{(p2^k)}$: $\mathbf{X}^{(p2^k)} \to \mathbf{Y}^{(p2^k)}$, which is the "parallel composition" of two networks $\mathscr{B}_{p,up}^{(p2^{k-1})}$: $\mathbf{X}_{up}^{(p2^{k-1})} \to \mathbf{Y}_{up}^{(p2^{k-1})}$ and $\mathscr{B}_{p,down}^{(p2^{k-1})}$: $\mathbf{X}_{down}^{(p2^{k-1})} \to \mathbf{Y}_{down}^{(p2^{k-1})}$;
- a bitonic merger network $\mathscr{BM}_p^{(p2^k)}$: $\mathbf{Z}^{(p2^k)} \to \mathbf{Y}^{(p2^k)}$.

Figure 2 depicts the network $\mathscr{B}_{12}$.

The following result has been shown following the paradigmatic methodology for proving correctness of balancing networks developed in [12] (see also Section 4).

THEOREM 3.3 ([10]). *The network* $\mathscr{BM}_p^{(p2^k)}$ *is a merging network.*

By construction of the network $\mathscr{B}_p^{(p2^k)}$, it immediately follows:

COROLLARY 3.4 ([10]). *The network* $\mathscr{B}_p^{(p2^k)}$ *is a counting network.*

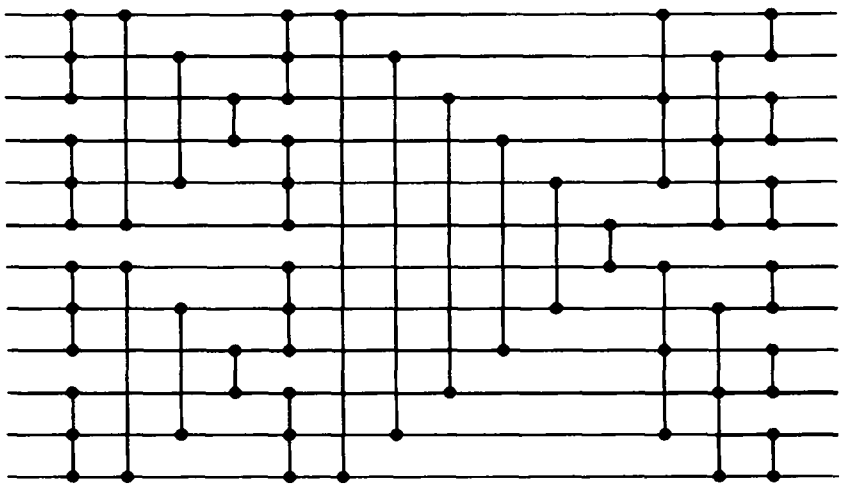

Fig. 2. The bitonic counting network $\mathscr{B}_{12}$.

It is trivial to establish that $depth(\mathscr{B}_p^{(p2^k)}) \in \Theta(k^2)$ and $size(\mathscr{B}^{(w)}) \in \Theta(2^k k^2)$. This amounts to a significant improvement over a corresponding construction by Aharonson and Attiya [2] (*not* based on the AKS network [4]) that achieves depth and size of $\Theta(k^3)$ and $\Theta(2^k k^3)$, respectively.

3.3. *A PERIODIC kp-SMOOTHING NETWORK WITH WIDTH p^k*

We describe a general construction of a kp-smoothing network $\mathscr{P}^{(p^k)}$, for any integers $p \geqslant 2$ and $k \geqslant 1$, following Hardavellas et al. [21]. This network uses p-balancers only and achieves depth $\Theta(k^2)$. The *periodic counting network* introduced by Aspnes et al. [6] is the special case where $p = 2$.

p-Chains and p-cochains. We present definitions for p-chains and p-cochains, patterned after those in [17] modified to accommodate p-balancers.

Consider a sequence $X = x_0, x_1, \ldots, x_{n-1}$. We represent each index (subscript) of a term in X as a *p-ary string*, i.e., a string over $\{0, 1, \ldots, p-1\}$ representing the index in the p-ary arithmetic system. In our discussion, we will use terms, indices, and representations of indices interchangeably.

DEFINITION 3.1. The *level* $=0$ *p-chain of* X is X itself. For any integer $i \geqslant 1$, a *level-i p-chain of* X is a subsequence of X whose indices have the same i low-order p-ary digits.

There are, for each integer $i \geqslant 0$, p^i level-i p-chains of X. For $i \geqslant 1$, the level-i p-chain of X corresponding to the i low-order p-ary digits $p_i p_{i-1} \cdots p_1$ will be denoted as $X^{p_i p_{i-1} \cdots p_1}$. In particular, $X^0, X^1, \ldots, X^{p-1}$ denote the level-1 p-chains of X, namely, the subsequences of X of terms with indices $0, 1, \ldots, p-1$ modulo p, respectively.

DEFINITION 3.2. Let $\mathscr{C}$ be a set of p-chains of X. The *p-cochain of* X *defined by* $\mathscr{C}$, $X^{\mathscr{C}}$, is the subsequence of X consisting of all terms of p-chains in $\mathscr{C}$.

Of special interest are p-cochains of X defined by certain combinations of level-2 p-chains of X. Specifically, let $\mathscr{C}_0 = \{X^{00}, X^{11}, \ldots, X^{p-1,p-1}\}$, $\mathscr{C}_1 = \{X^{01}, X^{12}, \ldots, X^{p-1,0}\}$, through $\mathscr{C}_{p-1} = \{X^{0,p-1}, X^{p-1,p-2}, \ldots, X^{10}\}$. Each of these sets of p-chains of X defines a corresponding p-cochain of X. These p-cochains, denoted by $X^{\mathscr{C}_0}$, $X^{\mathscr{C}_1}$, through $X^{\mathscr{C}_{p-1}}$, will be called the *special p-cochains of* X. Notice that, by definition of special p-cochains, $X^{ij} \in \mathscr{C}_k$ if and only if $j \equiv (i+k) \bmod p$.

As an example, consider the sequence $X = x_0, x_1, \ldots, x_{63}$. It is straightforward to obtain the special 4-cochains of X, namely, $X^{\mathscr{C}_0}$, $X^{\mathscr{C}_1}$, $X^{\mathscr{C}_2}$, and $X^{\mathscr{C}_3}$:

- $\mathscr{C}_0 = \{X^{00}, X^{11}, X^{22}, X^{33}\}$, and:

$$X^{\mathscr{C}_0} = x_0, x_5, x_{10}, x_{15}, x_{16}, x_{21}, x_{26}, x_{31}, x_{32}, x_{37}, x_{42}, x_{47}, x_{48}, x_{53}, x_{58}, x_{63}.$$

- $\mathscr{C}_1 = \{X^{01}, X^{12}, X^{23}, X^{30}\}$, and:

$$X^{\mathscr{C}_1} = x_1, x_6, x_{11}, x_{12}, x_{17}, x_{22}, x_{27}, x_{28}, x_{33}, x_{38}, x_{43}, x_{44}, x_{49}, x_{54}, x_{59}, x_{60}.$$

- $\mathscr{C}_2 = \{X^{02}, X^{20}, X^{13}, X^{31}\}$, and:

$$X^{\mathscr{C}_2} = x_2, x_7, x_8, x_{13}, x_{18}, x_{23}, x_{24}, x_{29}, x_{34}, x_{39}, x_{40}, x_{45}, x_{50}, x_{55}, x_{56}, x_{61}.$$

- $\mathscr{C}_3 = \{X^{03}, X^{32}, X^{21}, X^{10}\}$, and:

$$X^{\mathscr{C}_2} = x_3, x_4, x_9, x_{14}, x_{19}, x_{20}, x_{25}, x_{30}, x_{35}, x_{36}, x_{41}, x_{46}, x_{51}, x_{52}, x_{57}, x_{62}.$$

The next two lemmas establish important properties of p-chains and p-cochains that will be used later. The first lemma reveals the close relationship between p-chains and p-cochains.

LEMMA 3.5 ([21]). *For each integer $i > 1$, a subsequence of X is a level-i p-chain of X if and only if it is a level-$i-1$ p-chain of one of the special p-cochains of X.*

The construction. Define the balancing network $\mathscr{BL}^{(p^k)}$, called *block*, as follows. When k is equal to 1, the network $\mathscr{BL}^{(p^k)}$ consists of a single p-balancer. For larger values of k, the network $\mathscr{BL}^{(p^k)}$ is constructed recursively. We start with p networks $\mathscr{BL}^{(p^{k-1})}$ denoted by $\mathscr{N}_0, \mathscr{N}_1$, through $\mathscr{N}_{p-1}$. Given an input $\mathbf{X}$, the input to $\mathscr{N}_i$ is $\mathbf{X}^{\mathscr{B}_i}$, where $0 \leqslant i \leqslant p-1$. Let $\mathbf{Y}^{\mathscr{B}_i}$ be the output sequence for the network $\mathscr{N}_i$, $i \in [p]$. The final stage of the network $\mathscr{BL}^{(p^k)}$ combines each p-tuple $y_j^{\mathscr{B}_0}, y_j^{\mathscr{B}_1}$, through $y_j^{\mathscr{B}_{p-1}}$ in a single p-balancer, yielding final outputs z_{pj}, z_{pj+1}, through z_{pj+p-1}, $0 \leqslant j \leqslant p^{k-1}-1$.

For example, the case $p=3$ and $k=3$ yields the balancing network $\mathscr{BL}^{(27)}$ shown in Figure 3.

The balancing network $\mathscr{P}^{(p^k)}$ is the cascade of k balancing networks $\mathscr{BL}^{(p^k)}$ joined so that the ith output wire of one is the ith input wire of the next, $0 \leqslant i \leqslant p^k - 1$. Figure 4 shows the balancing network $\mathscr{P}^{(27)}$.

Through a series of lemmas providing interesting combinatorial properties of p-chains and p-cochains, it is shown:

THEOREM 3.6 ([21]). *The network $\mathscr{P}^{(p^k)}$ is a pk-smoothing network.*

Hardavellas et al. provide counterexample executions to show that for several values of $p \geqslant 3$, the network $\mathscr{P}^{(p^k)}$ is *not* a counting network. However, it turns out that the special case where $p=2$ is different:

THEOREM 3.7 ([6]). *The network $\mathscr{P}^{(2^k)}$ is a counting network.*

It is an important open question to precisely determine those values of p other than 2, if any, for which the network $\mathscr{P}^{(p^k)}$ is a counting network.

It is trivial to establish that $\mathit{depth}(\mathscr{P}^{(p^k)}) \in \Theta(k^2)$ and $\mathit{size}(\mathscr{P}^{(p^k)}) \in \Theta(p^k k^2)$.

3.4. A LOGARITHMIC-DEPTH COUNTING NETWORK

We describe an inductive construction of the *logarithmic-depth* counting network $\mathscr{S}_{t,w}$ with t and w input and output wires, respectively. The construction follows that presented by Busch and Mavronicolas [13]; it uses the *bounded-difference δ-merging network* $\mathscr{M}_w(\delta)$, whose construction is described next, as a basic module.

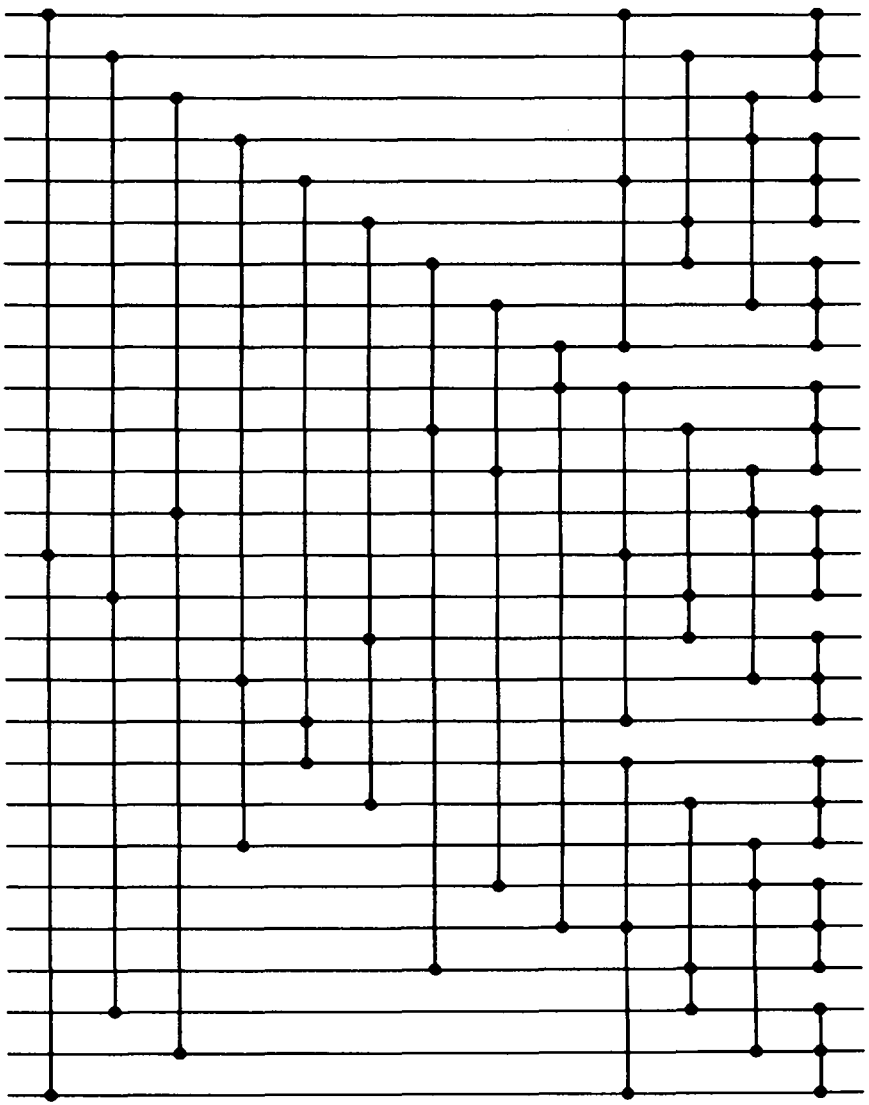

Fig. 3. The balancing network $\mathscr{BL}^{(27)}$.

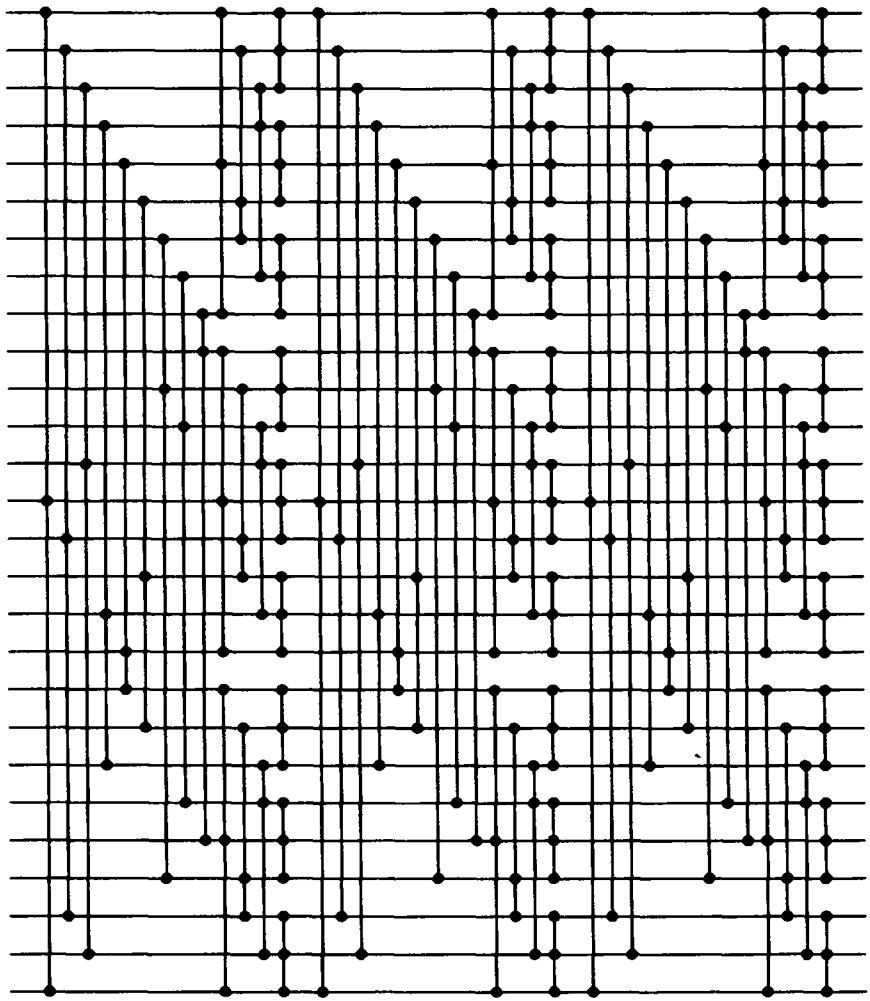

Fig. 4. The balancing network $\mathscr{P}^{(27)}$.

The bounded-difference δ-merging network $\mathscr{M}_w(\delta)$. We present a construction of a network that merges two input step sequences with a bounded difference between their sums into a sequence that has the step property. Assume, as usual, that w and δ are powers of 2 and $2 \leqslant \delta \leqslant w/2$, and consider a partition of $[w]$ into blocks π_0 and π_1. We start with a formal definition of this formal property.

A *bounded-difference δ-merging network* is a balancing network $\mathscr{B}$: $\mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ such that, if for every $r \in \{0,1\}$, then, for any i and k, $i,k \in \pi_r$ and $i<k$, $0 \leqslant x_i - x_k \leqslant 1$ and $\sum_{j \in \pi_0} x_j - \sum_{j \in \pi_1} x_j \leqslant \delta$, for any j and l, $0 \leqslant j < l \leqslant w-1$, $0 \leqslant y_j - y_l \leqslant 1$. That is, the input sequence is partitioned into two blocks, each of size $w/2$, and the output has the step property whenever the difference of sums of inputs in these blocks is at most δ, and the restrictions of the input sequence to each of these blocks each have the step property. The construction of a bounded-difference δ-merging network will involve, as a building block, a network that halves an upper bound on the difference between sums of inputs in each of two blocks that individually have the step property. A formal definition of this property follows.

A *bounded-difference δ-halving network* is a balancing network $\mathscr{B}$: $\mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ such that if for every $r \in \{0,1\}$, for any i and k, $i,k \in \pi_r$ and $i<k$, $0 \leqslant x_i - x_k \leqslant 1$, and $\sum_{j \in \pi_0} x_j - \sum_{j \in \pi_1} x_j \leqslant \delta$, then, for any j and l, $j,l \in \pi_r$ and $j<l$, $0 \leqslant y_j - y_l \leqslant 1$, and $\sum_{j \in \pi_0} y_j - \sum_{j \in \pi_1} y_j \leqslant \delta/2$. That is, the input and output sequences are each partitioned into two blocks, each of size $w/2$, and the difference of sums of outputs in these blocks is at most $\delta/2$, and the restrictions of the output sequence to each of these blocks has the step property whenever the difference of sums of inputs in these blocks is at most δ, and the restrictions of the input sequence to each of these blocks each have the step property. We present a bounded-difference δ-halving network $\mathscr{K}_w(\delta)$: $\mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$, which is just a single layer, consisting only of two-input balancers.

Let $\pi_0 = \{2k \mid 0 \leqslant k \leqslant w/2-1\}$ and $\pi_1 = \{2k+1 \mid 0 \leqslant k \leqslant w/2-1\}$; that is, π_0 and π_1 index the even and odd, respectively, subsequences of $\mathbf{X}^{(w)}$ and $\mathbf{Y}^{(w)}$. We describe the connections of $\mathscr{K}_w(\delta)$:

- For each k, $0 \leqslant k \leqslant \delta/2-1$, input wires $2k \in \pi_0$ and $2(w/2-\delta/2+k)+1 \in \pi_1$ are connected through a two-input balancer. The top and bottom output wires of this balancer are the $2k$th and $(2(w/2-\delta/2+k)+1)$th, respectively, output wires of the network $\mathscr{K}_w(\delta)$.
- For each k, $\delta/2 \leqslant k \leqslant w/2-1$, input wires $2k \in \pi_0$ and $2(k-\delta/2)+1 \in \pi_1$ are connected through a two-input balancer. The top and bottom output wires of this balancer are the $(2(k-\delta/2)+1th$ and $2k$th, respectively, output wires of the network $\mathscr{K}_w(\delta)$.

As an example, Figure 5 depicts the network $\mathscr{K}_8(4)$: $\mathbf{X}^{(8)} \to \mathbf{Y}^{(8)}$, with wires drawn as horizontal lines and balancers stretched vertically.

The bounded-difference δ-merging network $\mathscr{M}_w(\delta)$ is the cascade of $\log\delta$ bounded-difference δ-halving networks $\mathscr{K}_w(\delta), \mathscr{K}_w(\delta/2), \ldots, \mathscr{K}_w(2)$ in this order.

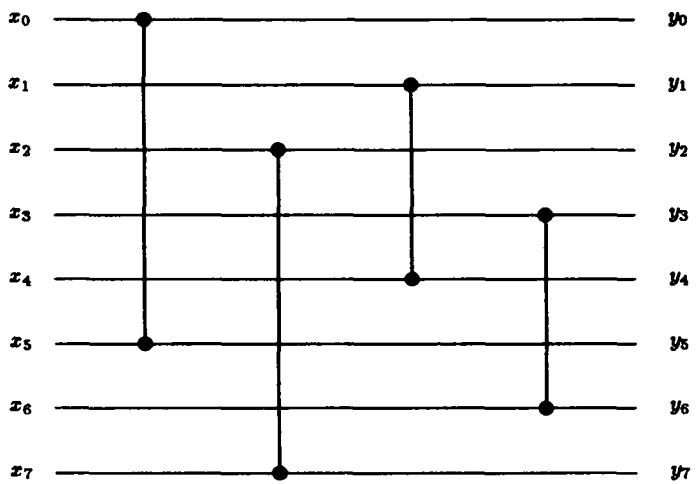

Fig. 5. The bounded-difference 4-halving network $\mathscr{H}_8(4)$.

A simple induction shows:

THEOREM 3.8 ([13]). *The network $\mathscr{M}^{(w)}(\delta)$ is a bounded-difference δ-merging network.*

The logarithmic-depth network $\mathscr{S}_{t,w}$. We continue with the construction, for any integers t and w which are powers of 2, $1 \leqslant t \leqslant w$ and $2 \leqslant w$, of the network $\mathscr{S}_{t,w}$.

For the construction, we proceed by induction on t. For the base case, we consider the cases $t = 1$ and $t = 2$ which are both special.

The case $t = 1$. For the first base case, where $t = 1$, we describe the construction of the network $\mathscr{S}_{1,w}$. We proceed by induction on w. For the base case, where $w = 2$, $\mathscr{S}_{1,2}$ consists of a single one-input balancer. Assume inductively that we have constructed $\mathscr{S}_{1,w/2}$ for some integer $w \geqslant 2$. We describe the construction of the network $\mathscr{S}_{1,w}$: $\mathbf{X}^{(1)} \to \mathbf{Y}^{(w)}$. Take a one-input balancer b. The input wire of the network $\mathscr{S}_{1,w}$ is the input wire to the balancer b. Take next two copies $\mathscr{S}^{[0]}_{1,w/2}$: $\mathbf{U}^{(1)} \to \mathbf{V}^{(w/2)}$ and $\mathscr{S}^{[1]}_{1,w/2}$: $\mathbf{W}^{(1)} \to \mathbf{Z}^{(w/2)}$ of $\mathscr{S}_{1,w/2}$. The top and bottom output wires of the balancer b are the input wires of the networks $\mathscr{S}^{[0]}_{1,w/2}$ and $\mathscr{S}^{[1]}_{1,w/2}$, respectively. Our inductive construction is completed by defining the output wires of the network $\mathscr{S}_{1,w}$ in terms of the output wires of the networks $\mathscr{S}^{[0]}_{1,w/2}$ and $\mathscr{S}^{[1]}_{1,w/2}$. The even and odd output subsequences of the network $\mathscr{S}_{1,w}$ are the output sequences of the networks $\mathscr{S}^{[0]}_{1,w/2}$ and

$\mathscr{S}^{[1]}_{1,w/2}$, respectively. Formally, for each i, $0 \leqslant i \leqslant w/2-1$, $y_{2i} = v_i$ and $y_{2i+1} = z_i$.

The case $t = 2$. For the second base case, where $t = 2$, we describe the construction of the network $\mathscr{S}_{2,w}$. We proceed by induction on w. For the base case, where $w = 2$, $\mathscr{S}_{2,2}$ consists of a single two-input balancer. Assume that we have constructed $\mathscr{S}_{1,w/2}$ for some integer $w \geqslant 2$. We describe the construction of the network $\mathscr{S}_{2,w}$: $\mathbf{X}^{(2)} \to \mathbf{Y}^{(w)}$. Take a two-input balancer b. The two input wires of the network $\mathscr{S}_{2,w}$ are the input wires to the balancer b. Take next two copies $\mathscr{S}^{[0]}_{1,w/2}$: $\mathbf{U}^{(1)} \to \mathbf{V}^{(w/2)}$ and $\mathscr{S}^{[1]}_{1,w/2}$: $\mathbf{W}^{(1)} \to \mathbf{Z}^{(w/2)}$ of $\mathscr{S}_{1,w/2}$. The top and bottom output wires of the balancer b are the input wires of the networks $\mathscr{S}^{[0]}_{1,w/2}$ and $\mathscr{S}^{[1]}_{1,w/2}$, respectively. Our construction is completed by defining the output wires of the network $\mathscr{S}_{2,w}$ in terms of the output wires of the networks $\mathscr{S}^{[0]}_{1,w/2}$ and $\mathscr{S}^{[1]}_{1,w/2}$. The even and odd output subsequences of the network $\mathscr{S}_{2,w}$ are the output sequences of the networks $\mathscr{S}^{[0]}_{1,w/2}$ and $\mathscr{S}^{[1]}_{1,w/2}$, respectively. Formally, for each i, $0 \leqslant i \leqslant w/2-1$, $y_{2i} = v_i$ and $y_{2i+1} = z_i$.

The case $t > 2$. Assume inductively that we have constructed the network $\mathscr{S}_{t/2,w/2}$ for all integers w and any integer t, $4 \leqslant t \leqslant w$. We describe the construction of the network $\mathscr{S}_{t,w}$. Take $t/2$ two-input balancers $b_0, b_1, \ldots, b_{t/2-1}$. For each i, $0 \leqslant i \leqslant t/2-1$, the input wires i and $i+t/2$ are the two input wires to the balancer b_i. Take next two copies $\mathscr{S}^{[0]}_{t/2,w/2}$: $\mathbf{U}^{(t/2)} \to \mathbf{V}^{(w/2)}$ and $\mathscr{S}^{[1]}_{t/2,w/2}$: $\mathbf{W}^{(t/2)} \to \mathbf{Z}^{(w/2)}$ of the network $\mathscr{S}_{t/2,w/2}$. For each i, $0 \leqslant i \leqslant t/2-1$, the top and bottom output wires of the balancer b_i are the ith input wires of the networks $\mathscr{S}^{[0]}_{t/2,w/2}$ and $\mathscr{S}^{[1]}_{t/2,w/2}$, respectively. Take next a bounded-difference $t/2$-merging network $\mathscr{M}_w(t/2)$. The output wires of the networks $\mathscr{S}^{[0]}_{t/2,w/2}$ and $\mathscr{S}^{[1]}_{t/2,w/2}$ are the even and odd subsequences, respectively, of the input sequence to the network $\mathscr{M}_w(t/2)$. The output sequence of the network $\mathscr{S}_{t,w}$ is the output sequence of the network $\mathscr{M}_w(t/2)$. Notice that since $t \geqslant 4$, the induction is grounded on $t = 2$.

As an example, Figure 6 depicts the counting network $\mathscr{S}_{4,16}$: $\mathbf{X}^{(4)} \to \mathbf{Y}^{(16)}$, using the same conventions as in Figure 5.

An inductive proof establishes:

THEOREM 3.9 ([13]). *The network* $\mathscr{S}_{t,w}$ *is a counting network.*

It is trivial to exploit the inductive construction of the network $\mathscr{S}_{t,w}$ and show that $depth(\mathscr{S}_{t,w}) = \log w + \log t(\log t - 1)/2 \in \Theta(\log w + \log^2 t)$, and $size(\mathscr{S}_{at,w}) = w - t + t \log t/2 + w \log t(\log t - 1)/4$.

We briefly comment on the extreme cases of values taken on by the parameter t. For $t = 1$ and $t = 2$, $\mathscr{S}_{1,w}$ and $\mathscr{S}_{2,w}$ are binary trees with

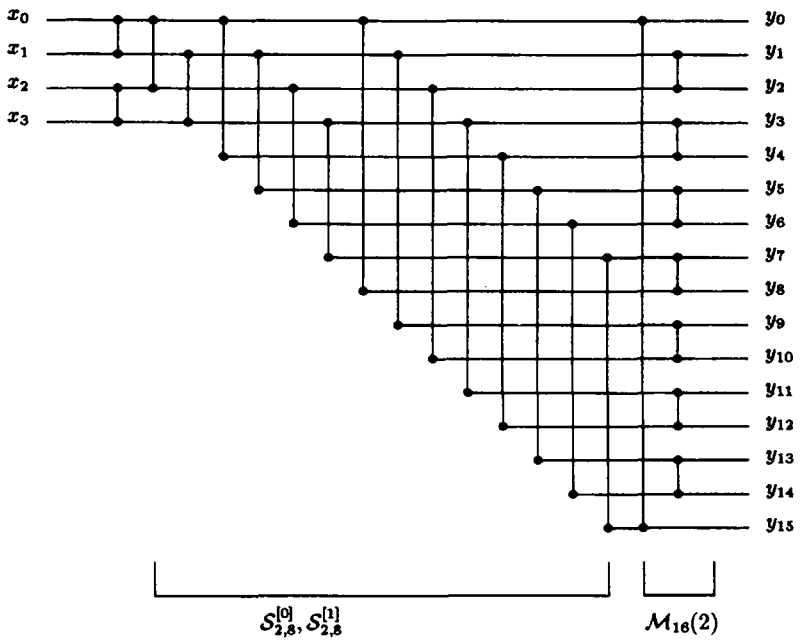

Fig. 6. The counting network $\mathcal{S}_{4,16}$.

$depth(\mathcal{S}_{1,w}) = depth(S_{2,w}) = \log w$ and $size(\mathcal{S}_{1,w}) = size(\mathcal{S}_{2,w}) = w - 1$. For $t = w$, $\mathcal{S}_{w,w}$ provides a counting-network alternative to the *bitonic* and *periodic* counting networks presented in [6]. This network attains identical bounds on depth and size with the bitonic and periodic networks, namely, $depth(\mathcal{S}_{w,w}) = \log w(\log w + 1)/2 \in \Theta(\log^2 w)$ and $size(\mathcal{S}_{w,w}) = w \log w(\log w + 1)/4 \in \Theta(w \log^2 w)$.

For suitably "small" t, the depth of the network $\mathcal{S}_{t,w}$ is $\Theta(\log w)$: *logarithmic* in the output width w. This is the first construction achieving logarithmic depth, and represents a vast improvement over previous constructions (e.g., [6, 26]).

4. COMBINATORIAL PROPERTIES

In this section, we provide a brief outline of a combinatorial theory of balancing networks developed by Busch and Mavronicolas [11], and its applications.

4.1. NOTATION

For any real number x, $\lceil x \rceil$ denotes the smallest integer no smaller than x, and $\lfloor x \rfloor$ denotes the largest integer no larger than x. For any vector $\mathbf{X}^{(w)} = \langle x_0, x_1, \ldots, x_w - 1 \rangle^{\mathrm{T}}$, $\lceil \mathbf{X}^{(w)} \rceil$ 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 $\mathbf{0}^{(w)}$ and $\mathbf{1}^{(w)}$ to denote the vectors $\langle 0, 0, \ldots, 0 \rangle^{\mathrm{T}}$ and $\langle 1, 1, \ldots, 1 \rangle^{\mathrm{T}}$, respectively, each with w entries. The definition of $\mathbf{1}^{(w)}$ is extended to matrices with w rows and w columns in the natural way to yield $\mathbf{1}^{(w \times w)}$. For any integer $p \geqslant 1$, denote $[p] = \{0, 1, \ldots, p-1\}$. We use $\mathbf{N}$ and $\mathfrak{R}$ to denote the sets of natural and real numbers, respectively.

The *minimum norm* function $\|\cdot\|_{\min}$: $\mathfrak{R}^w \to \mathfrak{R}$ is defined by $\|\mathbf{X}^{(w)}\|_{\min} = \min_{i \in [w]} |x_i|$. The *maximum norm* function $\|\cdot\|_{\max}$: $\mathfrak{R}^w \to \mathfrak{R}$ is defined by $\|\mathbf{X}^{(w)}\|_{\max} = \max_{i \in [w]} |x_i|$. Both the minimum and the maximum norms can be extended from vectors to matrices in the natural way. We will also use an extension of the maximum norm function from vectors to vector functions $\mathbf{F}$: $\mathbf{D} \to \mathfrak{R}^w$ over any domain $\mathbf{D}$, which is defined by setting $\|\mathbf{F}\|_{\max} = \max_{x \in \mathbf{D}} \|\mathbf{F}(x)\|_{\max}$; that is, $\|\mathbf{F}\|_{\max}$ is the maximum value attained by a component of $\mathbf{F}$ over the domain $\mathbf{D}$ of $\mathbf{F}$. The *1-norm* function $\|\cdot\|_1$: $\mathfrak{R}^w \to \mathfrak{R}$ is defined by $\|\mathbf{X}^{(w)}\|_1 = \sum_{i=0}^{w-1} |x_i|$.

Fix any integer $p \geqslant 2$, and let $\sum_{i \geqslant 0} x_i p^i$ denote the representation of the integer $x \geqslant 0$ in the p-ary arithmetic system, where, for each i, $x_i \in [p]$. For any integer $k \geqslant 1$, define $x \downarrow_p k = \sum_{0 \leqslant i \leqslant k-1} x_i p^i$ and $x \uparrow_p k = \sum_{i \geqslant k} x_i p^i$; that is, $x \downarrow_p k$ is the integer represented by the k least significant 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, $x \downarrow_p k + x \uparrow_p k = x$. The following result provides simple expressions for $x \downarrow_p k$ and $x \uparrow_p k$.

LEMMA 4.1. *For any integers* $x \geqslant 0$, $p \geqslant 2$, *and* $k \geqslant 1$,

$$x \uparrow_p k = \left\lfloor \frac{x}{p^k} \right\rfloor p^k \quad \textit{and} \quad x \downarrow_p k = x - \left\lfloor \frac{x}{p^k} \right\rfloor p^k.$$

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 $\mathbf{X}^{(w)}$ to yield

$$\mathbf{X}^{(w)} \downarrow_p k = \langle x_0 \downarrow_p k, x_1 \downarrow_p k, \ldots, x_{w-1} \downarrow_p k \rangle^{\mathrm{T}},$$

$$\mathbf{X}^{(w)} \uparrow_p k = \langle x_0 \uparrow_p k, x_1 \uparrow_p k, \ldots, x_{w-1} \uparrow_p k \rangle^{\mathrm{T}},$$

$$\mathbf{X}^{(w)} \updownarrow_p k = \langle x_0 \updownarrow_p k, x_1 \updownarrow_p k, \ldots, x_{w-1} \updownarrow_p k \rangle^{\mathrm{T}}.$$

In all of our discussion, we will refer to a set $\mathscr{P} = \{p_0, p_1, \ldots, p_{m-1}\}$ of positive integers no less than 2, and we will let P denote the least common multiple of integers in $\mathscr{P}$. Without loss of generality, assume p_0 is the largest integer in $\mathscr{P}$.

4.2. A MATRIX REPRESENTATION

In case $depth(\mathscr{B}) = 1$, $\mathscr{B}$ will be called a *layer*, and it will be uniquely represented by a matrix $\mathbf{I}_{\mathscr{B}}$ with w rows and w columns, called the *incidence matrix*,[2] which determines incidences between input and output wires, and a vector $\mathbf{O}_{\mathscr{B}}$ with w rows, called the *order vector*, which determines the order of each output wire. Formally, we define:

- for any i and j, $0 \leqslant i, j \leqslant w-1$, $\mathbf{I}_{\mathscr{B}}[ji] = 1/p$ if input wire i and output wire j are connected via a p-balancer, for some $p \in \mathscr{P}$; else $\mathbf{I}_{\mathscr{B}}[ji] = 1$ if output wire j coincides with input wire i, and 0 otherwise;
- for any j, $0 \leqslant j \leqslant w-1$, $\mathbf{O}_{\mathscr{B}}[j] = \mathrm{ord}(j)$ if output wire j is the output wire of a balancer; else $\mathbf{O}_{\mathscr{B}}[j] = 0$.

For example, for the layer $\mathscr{B}$ depicted in Figure 7 using the same conventions as for Figure 1, we have that

$$\mathbf{I}_{\mathscr{B}} = \begin{pmatrix} 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 1/3 & 1/3 & 1/3 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \end{pmatrix},$$

[2] Our notion of an incidence matrix is different from all similar notions of incidence matrices encountered in combinatorial matrix theory (see, e.g., [9]) in that it explicitly uses node degrees.

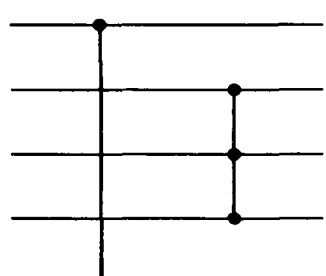

Fig. 7. The layer $\mathscr{B}$.

and

$$\mathbf{O}_{\mathscr{B}} = \langle 0 \quad 0 \quad 1/3 \quad 2/3 \quad 1/2 \rangle^{\mathrm{T}}.$$

Apparently, by definitions of balancers, the incidence matrix $\mathbf{I}_{\mathscr{B}}$, and the order vector $\mathbf{O}_{\mathscr{B}}$, it immediately follows:

PROPOSITION 4.2 ([11]). *For a layer* $\mathscr{B}$: $\mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$,

$$\mathbf{Y}^{(w)} = \left[\mathbf{I}_{\mathscr{B}} \cdot \mathbf{X}^{(w)} - \mathbf{O}_{\mathscr{B}}\right].$$

If $depth(\mathscr{B}) = d > 1$, then $\mathscr{B}$ can be uniquely partitioned into layers $\mathscr{B}_1, \mathscr{B}_2, \ldots, \mathscr{B}_d$ from left to right in the obvious way. The incidence matrix $\mathbf{I}_{\mathscr{B}_i}$ and the order vector $\mathbf{O}_{\mathscr{B}_i}$ are associated with layer $\mathscr{B}_i$, $1 \leqslant i \leqslant d$. We represent $\mathscr{B}$ by the sequence of d connection matrices $\mathbf{I}_{\mathscr{B}_1}, \mathbf{I}_{\mathscr{B}_2}, \ldots, \mathbf{I}_{\mathscr{B}_d}$, and the sequence of d order vectors $\mathbf{O}_{\mathscr{B}_1}, \mathbf{O}_{\mathscr{B}_2}, \ldots, \mathbf{O}_{\mathscr{B}_d}$.

4.3. A STRUCTURAL RESULT

Busch and Mavronicolas [11] show that for any balancing network, the outputs take a particular algebraic form as a function of the inputs, depending on the types of balancers used, and the depth and topology of the network.

THEOREM 4.3 ([11]). *Let* $\mathscr{B}$: $\mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ *be a balancing network of depth* d *over* $\mathscr{P}$ *with associated incidence matrices* $\mathbf{I}_{\mathscr{B}_1}, \mathbf{I}_{\mathscr{B}_2}, \ldots, \mathbf{I}_{\mathscr{B}_d}$ *and order vectors* $\mathbf{O}_{\mathscr{B}_1}, \mathbf{O}_{\mathscr{B}_2}, \ldots, \mathbf{O}_{\mathscr{B}_d}$. *Then*

$$\mathbf{Y}^{(w)} = \mathbf{C}_{\mathscr{B}} \cdot \mathbf{X}^{(w)} \uparrow_2 d + \mathbf{F}_{\mathscr{B}}\left(\mathbf{X}^{(w)} \downarrow_2 d\right),$$

for some matrix $\mathbf{I}_{\mathscr{B}}$ *and vector function* $\mathbf{F}_{\mathscr{B}}$: $[2^d]^w \to \mathbf{N}^w$, *such that*

(1) $\mathbf{I}_{\mathscr{B}} = \mathbf{I}_{\mathscr{B}_d} \cdot \mathbf{I}_{\mathscr{B}_{d-1}} \cdot \cdots \cdot \mathbf{I}_{\mathscr{B}_1}$, *and*
(2) $\mathbf{F}_{\mathscr{B}} = \mathbf{F}_{\mathscr{B}_d}$, *where the vector functions* $\mathbf{F}_{\mathscr{B}_l}$: $[P^l]^w \to \mathbf{N}^w$, $1 \leqslant l \leqslant d$, *are defined recursively as follows*:

$$\mathbf{F}_{\mathscr{B}_l}\left(\mathbf{X}^{(w)} \downarrow_P l\right) = \begin{cases} \left[\mathbf{I}_{\mathscr{B}_l} \cdot \mathbf{I}_{\mathscr{B}_{l-1}} \cdot \cdots \cdot \mathbf{I}_{\mathscr{B}_1} \cdot \mathbf{X}^{(w)} \updownarrow_P l \right. \\ \left. + \mathbf{I}_{\mathscr{B}_l} \cdot \mathbf{F}_{\mathscr{B}_{l-1}}\left(\mathbf{X}^{(w)} \downarrow_P (l-1)\right) - \mathbf{O}_{\mathscr{B}_l}\right], & l > 1, \\ \left[\mathbf{I}_{\mathscr{B}_1} \cdot \mathbf{X}^{(w)} \updownarrow_P 1 - \mathbf{O}_{\mathscr{B}_1}\right], & l = 1. \end{cases}$$

The proof of Theorem 4.3 follows a straightforward induction on the depth d of the balancing network $\mathscr{B}$.

Call the matrix $\mathbf{I}_{\mathscr{B}}$ the *steady transfer matrix of* $\mathscr{B}$. *Call the vector function* $\mathbf{F}_{\mathscr{B}}$ *the transient transfer function of* $\mathscr{B}$. *Call* $\mathbf{I}_{\mathscr{B}}$ *and* $\mathbf{F}_{\mathscr{B}}$ *the transfer parameters of* $\mathscr{B}$.

Theorem 4.3 shows that the output vector of a balancing network is the sum of two terms. The first term $\mathbf{I}_{\mathscr{B}} \cdot \mathbf{X}^{w} \uparrow_{P} d$, called the *steady output* term, involves the most significant part $\mathbf{X}^{(w)} \uparrow_{P} d$ of the input vector; this part is obtained by setting the d least significant P-ary digits of each entry of the input vector to zero. The steady output term is a *linear* transformation, defined by the steady transfer matrix $\mathbf{I}_{\mathscr{B}}$, of the most significant part of the input vector.

The second term $[\mathbf{I}_{\mathscr{B}} \cdot \mathbf{X}^{(w)} \updownarrow_{P} d + \mathbf{I}_{\mathscr{B}_d} \cdot \mathbf{F}_{\mathscr{B}_{d-1}}(\mathbf{X}^{(w)} \downarrow_{P} (d-1)) - \mathbf{O}_{\mathscr{B}_d}]$, called the *transient output* term, involves the least significant part $\mathbf{X}^{(w)} \downarrow_{P} d$ of the input vector; this part corresponds to the d least significant P-ary digits of each entry of the input vector. The transient output term is the image, under the transient transfer function $\mathbf{F}_{\mathscr{B}}$ of $\mathscr{B}$, of the least significant part of the input vector; apparently, the least significant part of the input vector undergoes a *nonlinear* transformation defined by $\mathbf{F}_{\mathscr{B}}$.

Thus, the steady transfer matrix $\mathbf{I}_{\mathscr{B}}$ is determined by the relative incidences in the network and shapes the steady output term, while the transient transfer function $\mathbf{F}_{\mathscr{B}}$ is determined by both the relative incidences in the network and the relative order of outputs for each balancer, and shapes the transient output term.

4.4. *COMBINATORIAL CHARACTERIZATIONS*

Busch and Mavronicolas [11] use Theorem 4.3 to obtain precise combinatorial characterizations for counting, K-smoothing, and merging networks. These characterizations are stated as necessary and sufficient conditions on the transfer parameters of a network. We start with a necessary and sufficient condition for a counting network.

THEOREM 4.4 ([11]). *The network* $\mathscr{B}$: $\mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ *is a counting network if and only if*:

(1) $\mathbf{I}_{\mathscr{B}} = (1/w)\mathbf{1}^{(w \times w)}$, *and*
(2) $\mathbf{F}_{\mathscr{B}}$ *is step on* $[P^{d}]^{w}$.

We continue with a corresponding combinatorial characterization for K-smoothing networks.

THEOREM 4.5 ([11]). *The network* $\mathscr{B}$: $\mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ *is a K-smoothing network if and only if*:

(1) $\mathbf{I}_{\mathscr{B}} = (1/w)\mathbf{1}^{(w \times w)}$, *and*
(2) $\mathbf{F}_{\mathscr{B}}$ *is K-smooth on* $[P^d]^w$.

Theorems 4.4 and 4.5 reveal that a counting (resp., K-smoothing) network distributes uniformly on its output wires the part of the inputs corresponding to the most significant digits, while the counting (resp., K-smoothing) property is inherited down to the network's response to the part of the inputs corresponding to the least significant digits. Most surprisingly, the next result suggests that this response is *insignificant* for smoothing networks.

THEOREM 4.6 ([11]). *The network* $\mathscr{B}$: $\mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ *is a smoothing network if and only if* $\mathbf{I}_{\mathscr{B}} = (1/w)\mathbf{1}^{(w \times w)}$.

Finally, Theorem 4.3 can be used to show a *conditional* combinatorial characterization for merging networks. For each integer k, denote by $(step_{\Pi}(\mathbf{N}^w))\downarrow_P k$ the set of all integer vectors $\mathbf{X}^{(w)} \in [P^k]^w$ such that $\mathbf{X}^{(w)} = \mathbf{Y}^{(w)}\downarrow_P k$ for some integer vector $\mathbf{Y}^{(w)} \in step_{\Pi}(\mathbf{N}^w)$; that is, $(step_{\Pi}(\mathbf{N}^w))\downarrow_P k$ is the set of the restrictions to their k least significant binary digits of input vectors with w entries that a merging network of output width w is required to count. We have:

THEOREM 4.7 ([11]). *For a network* $\mathscr{B}$: $\mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$, *assume* $\mathbf{I}_{\mathscr{B}}[ji] = 1/w$ *for all* $i, j \in [w]$. *Then,* $\mathscr{B}$ *is a merging network if and only if the vector function* $\mathbf{F}_{\mathscr{B}}$ *is step on* $(step_{\Pi}(\mathbf{N}^w))\downarrow_P d$.

Theorem 4.7 reveals that in some cases a merging network does actually do more than what its formal definition requires; more specifically, if the steady transfer matrix of a merging network is a constant matrix, then the merging network produces a step output vector on an input vector which is *not* step, has all of its entries no more than 2^{d-1} (i.e., each of its entries can be represented with d binary digits), but can be extended to a step vector by "sticking" most significant binary digits to the left of each of its entries.

Theorems 4.4, 4.5, and 4.6 suggest corresponding methodologies for proving correctness of counting, K-smoothing, and smoothing networks, respectively. More specifically, to show that a balancing network $\mathscr{B}$ is a counting, K-smoothing, or smoothing network, one computes expressions for the transfer parameters $\mathbf{I}_{\mathscr{B}}$ and $\mathbf{F}_{\mathscr{B}}$ and verifies inductively that the necessary and sufficient conditions involved in Theorems 4.4, 4.5, and 4.6 hold. A corresponding methodology for proving correctness of merging networks applies to the class of merging networks satisfying the combina-

torial condition on the steady transfer matrix assumed in Theorem 4.7. Concrete applications of this general methodology on specific examples of merging networks appear in [12, 10] (see also Sections 3.1 and 3.2), where it is shown that the bitonic merger network and a generalization of it actually do something stronger than merging.

5. IMPOSSIBILITY RESULTS

Sections 5.1 and 5.2 include impossibility results for constructible widths and lower bounds on size, respectively, for various classes of balancing networks.

For a collection $\mathscr{P} = \{p_0, p_1, \ldots, p_{m-1}\}$ of positive integers no less than 2, a balancing network $\mathscr{B}$ is a *balancing network over* $\mathscr{P}$ if it uses, for each $p \in \mathscr{P}$, a p-balancer. Set $p_0 = \max_{p \in \mathscr{P}} p$ and P to be the least common multiple of $p_0, p_1, \ldots, p_{m-1}$.

5.1. CONSTRUCTIBLE WIDTHS

The first impossibility results for balancing networks have been presented by Aharonson and Attiya [2].

THEOREM 5.1 ([2]). *Assume there exists a prime factor f of w such that, for each $p \in \mathscr{P}$, f does not divide p. Then, for any integer $K \geqslant 1$, there is no K-smoothing network of output width w over $\mathscr{P}$.*

Theorem 5.1 has been shown using a lemma which is a special case of the general Theorem 4.3.

Since for any integer $K \geqslant 1$, a counting network is also a K-smoothing network, Theorem 5.1 immediately implies:

COROLLARY 5.2 ([2]). *Assume there exists a prime factor f of w such that, for each $p \in \mathscr{P}$, f does not divide p. Then, there is no counting network of output width w over $\mathscr{P}$.*

For counting and K-smoothing networks of depth d over $\mathscr{P}$, Busch and Mavronicolas [11] show that the only constructible widths are the divisors of P^d.

THEOREM 5.3 ([11]). *Assume $\mathscr{B}$: $\mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ is a counting or K-smoothing network of depth d over $\mathscr{P}$. Then, w divides P^d.*

We remark that the proof of Theorem 5.3 [11] relies on a property of the steady transfer matrix that is necessary for smoothing networks, but not on any property of the transient transfer function. This suggests that

width limitations are, in general, consequences of the steady response of a balancing network. Notice also that the necessary condition in Theorem 5.3 does not involve the constant K.

As for K-smoothing networks, Theorem 5.3 strictly strengthens Theorem 5.1. We argue that Theorem 5.3 is indeed strictly stronger. Let the unique prime factorization of w be $w = \prod w_i^{l_i}$. Since P is the least common multiple of integers in $\mathscr{P}$, the unique prime factorization of P is $P = \prod p_i^{\max_p l_i(p)}$, where the product is taken over all prime factors p_i of integers $p \in \mathscr{P}$, and $l_i(p)$ is the degree of p_i in the unique prime factorization of p. Assume first that w divides P^d. Then, clearly, each prime factor of w is equal to some p_i in the unique prime factorization of P; hence, this prime factor divides each $p \in \mathscr{P}$ such that p_i is a prime factor of p. Assume now that for each prime factor w_i of w, there exists some $p \in \mathscr{P}$ such that w_i divides p. Clearly, w_i is equal to some p_i in the unique prime factorization of P. However, w may not divide P^d if $l_i > d \max_p l_i(p)$. Moreover, Theorem 5.3 is the generalization to an arbitrary set of balancer types of a corresponding impossibility result for K-smoothing networks over $\{2\}$ that has been claimed in [29], namely, that w divides 2^d.

5.2. *LOWER BOUNDS*

We start with a general lower bound on the distance between any given pair of input and output wires in a balancing network under a condition on the corresponding entry of the incidence matrix.

THEOREM 5.4 ([11]). *For a balancing network* $\mathscr{B}$: $\mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ *over* $\mathscr{P}$, *assume, for any indices* $i, j \in [w]$, *that* $\mathbf{I}_{\mathscr{B}}[ji] = 1/w$. *Then,* $dist_{\mathscr{B}}(i, j) \geq \log_{p_0} w$.

By Theorems 4.4 and 4.5, for any counting or K-smoothing network, $\mathbf{C}_{\mathscr{B}}[ji] = 1/w$ for all $i, j \in [w]$. Hence, Theorem 5.4 implies:

COROLLARY 5.5 ([11]). *Assume* $\mathscr{B}$: $\mathbf{X}^{(w)} \to \mathbf{Y}^{(w)}$ *is a counting or* K-*smoothing network over* $\mathscr{P}$. *Then, for all* $i, j \in [w]$, $dist_{\mathscr{B}}(i, j) \geq \log_{p_0} w$.

Corollary 5.5 implies that, for counting or K-smoothing networks, *every* path from an input wire to an output wire must have length at least $\log_{p_0} w$. In [6, Corollary 2.5], it is shown that the depth of any counting network of width w over $\{2\}$ is at least $\log_2 w$; i.e., there exists *some* path from an input wire to an output wire of length at least $\log_2 w$. Clearly, Corollary 5.5 strictly strengthens and generalizes this to an arbitrary set of

balancer types. Moreover, Corollary 5.5 represents a corresponding improvement to an observation in [29, Section 5] that the depth of a K-smoothing network over $\{2\}$ is at least $\log_2 w$.

6. PERFORMANCE

In this section, we survey results on evaluating the performance of balancing networks by both theoretical and experimental means.

Given the practical motivations that led to the invention of balancing networks, it is quite natural that the work introducing balancing networks [6] already provides experimental evidence that balancing networks significantly outperform conventional synchronization techniques under a variety of circumstances. More specifically, Aspnes et al. [6] analyze the *throughput* of counting networks for computations in which tokens are eventually spread through the network; they compare the performance of several implementations of shared counters, producers/consumers buffers, and barrier synchronization on a shared-memory multiprocessor. Aspnes et al. report that under sufficiently high levels of concurrency, implementations based on the bitonic and periodic networks outperform conventional implementations based on spin locks, sometimes dramatically.

Herlihy, Lim, and Shavit [22] take the next step in studying the actual performance of balancing networks. Arguing that dynamic load balancing can have a dramatic effect on the performance of parallel programs, Herlihy et al. investigate the performance of basic techniques for dynamic load balancing on large-scale multiprocessors; more specifically, they consider concurrent data structures based on: (1) spin locks with exponential backoff [1], (2) "queue" locks [28], (3) software combining trees [20], and (4) the bitonic counting network of Aspnes et al. (Section 3.1). Herlihy et al. run a series of simple benchmarks on a simulated 64-processor Alewife machine, a distributed-memory multiprocessor currently under development at MIT. This machine supports the shared-memory programming model. Although the two locking techniques are known to perform well on small-scale, bus-based multiprocessors, Herlihy et al. found that they are severely affected by contention as concurrency increases; they report that both the bitonic counting networks and combining trees have the same scaling behavior: they are more susceptible to variations in the interarrival times of increment requests because they hold locks for long durations. This is so because two requests arriving at a node must arrive within a small time window for combining to occur. Additionally, locks that are held for a significant amount of time at the combining tree nodes may block progress up the tree.

Dwork, Herlihy, and Waarts [18] pursue the first formal study of performance for balancing networks. Dwork et al. introduce for the first time a formal complexity model for contention in shared-memory multiprocessors; they consider a more general setting of a multiple instruction/multiple data (MIMD) architecture in which n asynchronous processes communicate by applying *read*, *write*, and *read-modify-write* operations to a shared memory.[3] Asynchrony means that there is no bound on processes' relative speeds. In real shared-memory multiprocessors, sources of asynchrony include page faults, cache misses, scheduling preemption, clock skew, variation in instruction speeds, and perhaps even processor failure.

In the model of Dwork et al., simultaneous accesses to a single memory location are serialized: only one operation succeeds at a time, and other pending operations must *stall*. The measure of contention introduced by Dwork et al. is the worst-case number of stalls that can be induced by an adversary scheduler. This model (like all complexity models) is an abstraction of how real machines actually behave. Nevertheless, it is believed to be accurate enough to make useful comparisons, and simple enough to be tractable. In particular, this model is well-suited for comparing alternative algorithms, and for deriving lower and upper bounds.

Dwork et al. use their model to derive tight or, in some cases, nearly tight asymptotic bounds on the contention produced by several classes of counting networks studied in the literature. In each case, they show that the contention in the counting network is substantially lower than the contention incurred by the conventional single-variable implementation of a shared counter. Experiments discussed earlier have provided evidence that certain counting networks outperform conventional single-variable counters at high levels of concurrency. The results of Dwork et al. [18] formally explain this phenomenon.

On a MIMD shared-memory multiprocessor machine, a balancing network is implemented as a shared-data structure, where balancers are records and wires are pointers from one record to another. Each of the machine's n asynchronous processors runs a program that repeatedly traverses the data structure from some input pointer to some output

[3] Recall that a read-modify-write operation atomically reads a value v from a memory location, writes back $f(v)$, where f is a predefined function, and returns v back to the caller. Nearly all modern processor architectures support some form of read-modify-write for interprocess synchronization. Common read-modify-write instructions include *test-and-set*, memory-to-register *swap*, *fetch-and-add*, *compare-and-swap*, and *load-linked/store-conditional* instructions.

pointer, each time shepherding a new token through the network. Tokens generated by processor p enter the network on input wire $p \bmod t$. The limitation on the number of concurrent processors implies a limitation on the number of tokens concurrently traversing the network at any given time: $\sum_{i=0}^{t-1} x_i - \sum_{j=0}^{w-1} y_j \leqslant n$. Consider an execution of a balancing network $\mathscr{B}$ entering a quiescent state after m tokens pass through it. Each time a token passes through a balancer, all tokens pending at this balancer incur a *stall* step, modeling their delay due to contention with each other. The number of stall steps has been introduced in [18] as a measure of contention. The *contention incurred by the traversal of m tokens through the network* $\mathscr{B}$ *at concurrency* n, denoted $cont(m, n, \mathscr{B})$, is the maximum number of stalls, over all possible executions, induced by an adversary scheduler. The *amortized contention of the network* $\mathscr{B}$ *at concurrency* n, denoted $cont(n, \mathscr{B})$, is the limit of $cont(m, n, \mathscr{B})$ divided by m, as m goes to infinity.

Dwork et al. [18] use rather ad hoc operational arguments, relying on execution patterns of the bitonic counting network, to show:

THEOREM 6.1 ([18]).

$$cont(n, \mathscr{B}^{(w)}) \in \Theta(n \log^2 w/w).$$

Dwork et al. [18] also claim:

THEOREM 6.2 ([18]).

$$cont(n, \mathscr{P}^{(2^k)}) \in O(nk^3/2^k).$$

Hardavellas et al. [21] introduce the so-called *recurrence relation method* for analyzing the contention of constructions of balancing networks. Roughly speaking, this method amounts to exploiting recursiveness in the construction of balancing networks in order to derive and solve a recurrence relation for contention. Hardavellas et al. apply their method on their periodic, kp-smoothing network and show:

THEOREM 6.3 ([21]).

$$cont(n, \mathscr{P}^{(p^k)}) \in \Theta(nk^2/2^k).$$

Notice that for $p = 2$, Theorem 6.3 improves Theorem 6.2.

For other instances where the recurrence relation method has been used, we quote:

THEOREM 6.4 ([10]). *For any* $k \geqslant 1$,

$$cont\left(n, \mathscr{B}_p^{(p2^k)}\right) \leqslant \frac{n}{2^{k-1}}\left(\frac{1}{p}\left(\frac{k(k+1)}{2}+k\right)+k\right)-\frac{k(k+3)}{2}.$$

THEOREM 6.5 ([13]).

$$cont(n, \mathscr{S}_{t,w}) \in \Theta(n \log t(1/t + \log t/w)).$$

Kapidakis and Mavronicolas [24], and, independently, Aiello et al. [3], generalize balancing networks to *load balancing networks* that accommodate jobs of varying completion times; they formalize criteria for the performance of load balancing networks and present and check two constructions against these criteria. Experimental results are also presented in [24], where load balancing networks are compared against a modification of the "queue" lock algorithm, studied in [22].

Finally, Shavit and Zemach [32] introduce *diffracting trees* as a new counting structure, consisting of a binary tree shown to be a counting network and a collection of "prisms," one in front of each balancer, serving to combine tokens in a randomized way in order to reduce memory contention. Since the construction is trivial, the only contribution of Shavit and Zemach lies in the idea of using "prisms". Nevertheless, this idea has not been exploited in full and no theoretical analysis has been provided for the expected behavior of prisms, even though Shavit and Zemach [32], and, more recently, Shavit and Touitou [31], present several astonishing experimental results on the actual performance of diffracting trees.

7. DISCUSSION AND FURTHER RESEARCH

Balancing networks deserve further study. We believe that they represent just a start toward a theory of low-contention data structures suitable for solving load balancing problems.

Work is still needed to derive lower and upper bounds, to develop other primitives, and to define new performance measures. Work is also needed in experimental directions, comparing balancing networks to other techniques for load balancing, for example, those based on exponential backoff

[1], and for understanding their behavior in architectures other than the single-bus architecture provided by Encore. (The results in [22] discussed in Section 6 that use the ASIM simulator of the MIT Alewife machine display a substantial gain in performance due to parallelism on such distributed-memory machines.)

Finally, potential applications of balancing networks in problems not seemingly of the balancing kind need to be explored. Attiya et al. [7] take the first step in this direction by showing how to use counting networks in devising asynchronous algorithms for the *lattice agreement* decision problem. We believe that further applications should be possible.

I am indebted to Costas Busch for sharing with me his invaluable insights into balancing networks during our collaboration. I am particularly thankful to Maurice Herlihy for his encouragement during all states of my research on balancing networks. Moreover, I have enjoyed many helpful and inspiring discussions on balancing networks with Hagit Attiya, Cynthia Dwork, Sarantos Kapidakis, Christos Nikolau, Gadi Taubenfeld, Nir Shavit, and Orli Waarts. This work has been supported by ESPRIT III Basic Research Project #8144 (LYDIA—Load Balancing on High-Performance Parallel and Distributed Systems). The author was partially supported by funds for the promotion of research at University of Cyprus (research project "Load Balancing Problems in Shared Memory Multiprocessor Architectures"). Part of this work was performed while the author was visiting Institute of Computer Science, Foundation for Research and Technology–Hellas.

REFERENCES

1. A. Agarwal and M. Cherian, Adaptive backoff synchronization techniques, in: *Proceedings of the 16th Annual ACM-IEEE International Symposium on Computer Architecture*, June 1989, pp. 396–406.
2. E. Aharonson and H. Attiya, Counting networks with arbitrary fan-out, in: *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, Jan. 1992, pp. 104–113.
3. W. Aiello, R. Venkatesan, and M. Yung, Coins, weights and contention in balancing networks, in: *Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing*, Aug. 1994, pp. 193–205.
4. M. Ajtai, J. Komlós, and E. Szemerédi, Sorting in $c \log n$ steps, *Combinatorica* 3:1–19 (1983).
5. T. E. Anderson, The performance of spin lock alternatives for shared-memory multiprocessors, *IEEE Trans. Paral. Distrib. Syst.* 1(1):6–16 (1990).
6. J. Aspnes, M. Herlihy, and N. Shavit, Counting networks, *J. ACM* 41(5):1020–1048 (1994).
7. H. Attiya, M. Herlihy, and O. Rachman, Atomic snapshots using lattice agreement, *Distrib. Comput.* 8:121–132 (1995).
8. C. Batcher, Sorting networks and their applications, in: *Proceedings of the AFIPS Spring Joint Computer Conference*, 1968, pp. 307–314.
9. R. A. Brualdi and H. J. Ryser, Combinatorial matrix theory, in: *Encyclopedia of Mathematics and Its Applications*, Vol. 21, Cambridge University Press, London/New York, 1991.

10. C. Busch, N. Hardavellas, and M. Mavronicolas, Contention in counting networks, in: *Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing*, Aug. 1994, p. 404.
11. C. Busch and M. Mavronicolas, A combinatorial treatment of balancing networks, in: *Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing*, Aug. 1994, pp. 206–215.
12. C. Busch and M. Mavronicolas, Proving correctness for balancing networks, in: P. M. Pardalos, K. G. Ramakrishnan, and M. G. C. Resende (Eds.), *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, Vol. 22, American Mathematical Society, Providence, RI, 1995, pp. 1–32.
13. C. Busch and M. Mavronicolas, A logarithmic-depth counting network, in: *Proceedings of the 14th Annual ACM Symposium on Principles of Distributed Computing*, Aug. 1995, p. 274.
14. C. Busch and M. Mavronicolas, Impossibility results for threshold networks, Technical Report FORTH-ICS/TR-137, Institute of Computer Science, Foundation for Research and Technology–Hellas, Sept. 1995.
15. C. Busch and M. Mavronicolas, Odd-even counting networks, unpublished manuscript.
16. T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, McGraw-Hill and MIT Press, New York and Cambridge, MA, 1990.
17. M. Dowd, Y. Perl, L. Rudolph, and M. Saks, The periodic balanced sorting network, *J. ACM* 36(4):738–757 (1989).
18. C. Dwork, M. Herlihy, and O. Waarts, Contention in shared memory algorithms, in: *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, May 1993, pp. 174–183.
19. E. W. Felten, A. LaMarca, and R. Ladner, Building counting networks from larger balancers, Technical Report 93-04-09, Dept. Computer Science and Engineering, University of Washington, Apr. 1993.
20. J. R. Goodman, M. K. Vernon, and P. J. Woest, Efficient synchronization primitives for large-scale cache-coherent multiprocessors, in: *Proceedings of the 3rd Annual ACM Symposium on Architectural Support for Programming Languages and Operating Systems*, Apr. 1989, pp. 64–75.
21. N. Hardavellas, D. Karakos, and M. Mavronicolas, Notes on sorting and counting networks, in: A. Schiper (Ed.), *Proceedings of the 7th International Workshop on Distributed Algorithms (WDAG-93), Lecture Notes in Computer Science*, Vol. 725 Springer-Verlag, Lausanne, Switzerland, 1993, pp. 234–248.
22. M. Herlihy, B.-C. Lim, and N. Shavit, Low contention load balancing on large-scale multiprocessors, in: *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, July 1992, pp. 219–227.
23. M. Herlihy, N. Shavit, and O. Waarts, Low contention linearizable counting networks, in: *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science*, Oct. 1991, pp. 526–535.
24. S. Kapidakis and M. Mavronicolas, Load balancing networks, in: *Proceedings of the 14th Annual ACM Symposium on Principles of Distributed Computing*, Aug. 1995, p. 275.
25. M. Klugerman, Small-depth counting networks and related topics, Ph.D. Thesis, Dept. Mathematics, Massachusetts Institute of Technology, Sept. 1994.
26. M. Klugerman and C. Plaxton, Small-depth counting networks, in: *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, May 1992, pp. 417–428.

27. D. Knuth, *Sorting and Searching, of The Art of Computer Programming*, Vol. 3, Addison-Wesley, Reading, MA, 1973.
28. J. M. Mellor-Crummey and M. L. Scott, Algorithms for scalable synchronization on shared-memory multiprocessors, Technical Report #342, Dept. Computer Science, University of Rochester, Apr. 1990.
29. S. Moran and G. Taubenfeld, A lower bound on wait-free counting, in: *Proceedings of the 12th Annual ACM Symposium on Principles of Distributed Computing*, Aug. 1993, pp. 251–259.
30. D. Peleg and E. Upfal, The token distribution problem, *SIAM J. Comput.* 18:229–241 (1989).
31. N. Shavit and D. Touitou, Elimination trees and the construction of pools and stacks, in: *Proceedings of the 7th Annual ACM Symposium on Parallel Algorithms and Architectures*, June 1995.
32. N. Shavit and A. Zemach, Diffracting trees, in: *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures*, June 1994, pp. 167–176.

Received 1 October 1995; revised 1 October 1996