Theoretical Computer Science 410 (2009) 3725–3732

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Accelerating certain outputs of merging and sorting networks

Tamir Levi *, Ami Litman

Faculty of Computer Science, Technion, Haifa 32000, Israel

ARTICLE INFO

Article history:
Received 1 October 2008
Accepted 28 April 2009
Communicated by G. Ausiello

Dedicated to the memory of Professor Shimon Even.

Keywords:
Merging
Sorting
Comparator networks
Zero–one principle
Acceleration

ABSTRACT

This work studies comparator networks in which several of the outputs are accelerated. That is, they are generated much faster than the other outputs, and this without hindering the other outputs. We study this acceleration in the context of merging networks and sorting networks.

The paper presents a new merging technique, the *Tri-section technique*, that separates, using a depth 1 network, two sorted sequences into three sets, such that every key in one set is smaller than or equal to any key in the following set. After this separation, each of these sets can be sorted separately, causing the above acceleration of certain outputs.

An additional contribution of this paper concerns the well-known 0–1 Principle [D.E. Knuth, The Art of Computer Programming vol. 3: Sorting and Searching, second edition, Addison-Wesley, 1998]. This principle is a powerful tool that simplifies the construction and analysis of comparator networks. The paper demonstrates that, in some cases, there is a better tool for achieving the same goal. In the case at hand, this new tool simplifies one of our proofs by having fewer special cases than the classical 0–1 Principle.

A second additional contribution concerns Batcher's merging techniques. It was shown in [T. Levy, A. Litman, On Merging Networks, Technical Report CS-2007-16, Technion, Department of Computer Science, 2007] that all published merging networks, whose width is a power of 2, are a natural generalization of Batcher's odd–even merging network. All these published merging networks are of minimal depth and have no degenerate comparators. This raises the following question. Is there a merging network, having the above properties, that is not a natural generalization of Batcher's odd–even merging network? The Tri-section technique provides a positive answer to this question.

1. Introduction

We study comparator networks in which several of the outputs are accelerated. That is, they are generated much faster than the other outputs, and this without hindering the other outputs. Namely, for every $0 < k \leq n$, we present a merging network of minimal depth that merges two sorted sequences of length n into a single sorted sequence. This merging network produces either the lowest k keys or the highest k keys[1] after a delay of $\lceil \log(k) \rceil + 1$ comparators. Building on that, we construct, for every $0 < k < n$, an n-key sorting network that accelerates its k lowest or its k highest outputs. This sorting network is a *merge-sort network*[2] and, apparently, is of minimal depth among these networks; this subject is further discussed in Section 6. Specifically, its depth is $\frac{\lceil \log(n) \rceil \cdot \lceil \log(2n) \rceil}{2}$, the same depth as for the Batcher merge-sort networks [2].

* Corresponding author. Tel.: +972 4 8293854.
E-mail addresses: levyt@cs.technion.ac.il (T. Levi), litman@cs.technion.ac.il (A. Litman).

1 When n is a power of 2, both the lowest k keys and the highest k keys can be accelerated.

2 A merge-sort network is a sorting network which operates as follows. The input is arbitrarily divided into two sets of (almost) equal size and each set is recursively sorted; the two sorted sequences are then merged.

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.04.024

However, in contrast to the Batcher merge-sort networks which may accelerate only the first and last outputs, our merge-sort networks accelerate either the k lowest keys or k highest keys (see footnote 1) to a delay of less than $\lceil \log(n) \rceil \cdot \lceil \log 2k \rceil$ comparators.

The paper presents a new merging technique, the *Tri-section technique*, that separates, using a depth 1 network, two sorted sequences into three sets, such that every key in one set is smaller than or equal to any key in the following set. After this separation, each of these sets can be sorted separately and this leads to the desired acceleration. The idea of separating the input into two sets is known and is used, for example, in the Bitonic sorter of Batcher [2]; however, to the best of our knowledge, separation into three sets as above is novel.

To put our results in context, let us compare the acceleration of our networks with the acceleration of other well-known merging networks — The Bitonic sorter and the odd–even merging network, both of Batcher [2,3]. The Bitonic sorter has no accelerated outputs at all; all outputs have exactly the same delay. On the other hand, the odd–even merging network has only two accelerated outputs, the first and the last ones whose delay is exactly 1. All other outputs have the same delay.

To the best of our knowledge, the idea of accelerating certain outputs has never been addressed. The only prior work which is somewhat similar to our work concerns *selectors*. A (k, n)-*selector* is a network that separates a set of n keys into the lowest k and the other keys. Fast selection leads to a sorting network that accelerates certain outputs, as follows: First, the k lowest keys are separated from the other keys. Afterwards, each set is sorted separately. Yao presented a (k, n)-selector which is efficient when k is constant and n is very large. This selector can be extended into a sorting network that accelerates its lowest k outputs; however, the depth of the resulting sorting network exceeds the minimal depth of a merge-sort network. Our network accelerates the lowest k outputs while its depth is minimal among merge-sort networks.

Our paper has several additional contributions. The first one concerns the well-known 0–1 Principle [6]. This principle is a powerful tool that simplifies the construction and analysis of comparator networks. The paper demonstrates that, in some cases, there is a more convenient tool for achieving the same goal. In the context of merging, we use a small and elegant set of vectors which constitute a *conclusive set* [4]; namely, a network is a merging network if and only if it sorts this set. This tool simplifies our proof by having fewer special cases than the classical 0–1 Principle.

The second additional contribution concerns Batcher's merging techniques. Batcher's odd–even merging technique [2] works as follows: Each of the input sequences is partitioned into its even part and its odd part. The even part of one sequence is merged with the even part of the other sequence recursively and, similarly, the odd parts are merged. Finally, the two resulting sequences are merged into a single sorted sequence by a depth 1 network.

A slight variant of this method, due to Knuth [6, pp 231] and Leighton [7, pp 623], recursively merges the even part of each input sequence with the odd part of the other sequence. Again, the resulting two sorted sequences can be merged by a depth 1 network. We refer to the family of networks produced by allowing each of the above two variants anywhere in the recursion process as *Batcher merging networks*. It was shown in [8] that all published merging networks, whose width is a power of 2, are members of this family. All these merging networks are of minimal depth and have no degenerate comparators. (A *degenerate*[3] comparator has a fixed incoming edge whose value is always greater than or equal to the value on the other incoming edge, for every valid input of the network.) The above fact raises the following question:

Question 1. *Are the Batcher merging networks the only merging networks with the following properties:*

1. *Their width, $2n$, is a power of 2.*
2. *Their depth is minimal* — $\log(2n)$.
3. *They have no degenerate comparators.*

The Tri-section technique provides a negative answer to this question, as shown in Section 5.

Another question, which remains open, concerns accelerating <u>all</u> the outputs of a merging network, each to a delay that is close to the trivial reachability bound of this output. This bound is due to the fact that the j lowest (or highest) output may come from each of certain $2j$ input edges. Therefore, our question is:

Question 2. *For any n (or arbitrary large n), is there a merging network of width $2n$ that, for every $j < n$, accelerates the j lowest output and the j highest output to a delay of* $\log(j) + o(\log(j))$*?*

2. Preliminaries

The concept of comparator networks is well-known and an example is depicted in Fig. 1. A comparator (represented by a circle) receives two keys via its two incoming edges. The comparator sorts these keys; it transmits the minimal one on the outgoing **Min** edge (indicated by a hollow arrowhead) and the maximal key on the outgoing **Max** edge (indicated by the solid arrowhead). The network's input edges are indicated by an open arrowhead.

[3] A degenerate comparator is a special case of a *redundant* comparator, studied in [8]. A comparator or a set of comparators is *redundant* if it can be removed from the network without disturbing the network's functionality. Clearly, a degenerate comparator is redundant but not the other way around. The above reference to a redundant set of comparators, rather than to a single redundant comparator, is crucial due to the following result of [8]. A merging network or a sorting network may have a set of redundant comparators while any single comparator cannot be removed without disturbing the network's functionality.

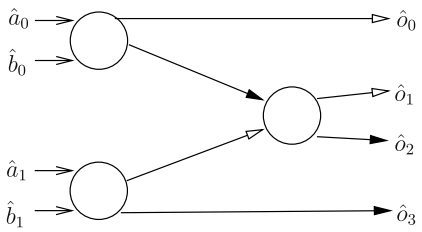

Fig. 1. A merging network of width 4 and depth 2.

The network of Fig. 1 is in fact a merging network. Its input is two sorted sequences, each of width 2 and its output is a sorted sequence of width 4. Keys enter the network through its *input edges* and exit the network through its *output edges*. We name the input edges to denote how the input is fed into the network. In such a network, one input sequence enters the edges $\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{n-1}$ and the other input sequence enters the edges $\hat{b}_0, \hat{b}_1, \ldots, \hat{b}_{n-1}$. Similarly, output edges are named $\hat{o}_0, \hat{o}_1, \ldots, \hat{o}_{2n-1}$ to denote how the output keys are assembled into a sequence. Namely, the output sequence $\vec{o} = \langle o_0, o_1, \ldots, o_{2n-1}\rangle$ is composed of the values on these edges, in that order.

The *width* of a network N is the number of its input edges which clearly equals the number of its output edges. Let e be an edge of N. The *depth* of e, denoted as $d(e)$, is the length of the longest path that ends in the tail of e (i.e., the path does not include e). Hence $d(e) = 0$ for every input edge e. The depth of N, denoted as $d(N)$, is the maximal depth of the edges of N. In the network, M, of Fig. 1, $d(M) = d(\hat{o}_1) = d(\hat{o}_2) = 2$ and $d(\hat{o}_0) = d(\hat{o}_3) = 1$.

A sequence of keys $\langle x_0, x_1, \ldots, x_{n-1}\rangle$ is denoted by $\vec{x}$; the *width* of this $\vec{x}$, denoted as $|\vec{x}|$, is n. A *bisequenced vector* is a pair of sequences of equal width and is denoted by $\langle \vec{a}, \vec{b}\rangle$. Naturally, $|\langle \vec{a}, \vec{b}\rangle| = 2|\vec{a}| = 2|\vec{b}|$. A *bisorted vector* is a bisequenced vector $\langle \vec{a}, \vec{b}\rangle$ in which both $\vec{a}$ and $\vec{b}$ are sorted. Such a vector is a valid input to a merging network of the appropriate width.

Let $\mathcal{K}$ be the set of optional keys. Usually the cardinality of $\mathcal{K}$ is insignificant, as long as it is greater than 1, and this paper is no exception. This fact is reflected by the 0–1 principle. Surprisingly, there are some properties of networks which depend on the size of $\mathcal{K}$ as shown in [10].

3. The Asymmetric Tri-section merging technique

This paper presents two Tri-section merging techniques. As we said, they are based on separating, using a depth 1 network, a bisorted vector of width $2n$ into three sequences, $\vec{x}, \vec{y}$ and $\vec{z}$, such that every key in one set is smaller than or equal to any key in the next set. This allows us to sort each of these sequences separately. In all of our techniques the resulting merging network is of a minimal depth. Furthermore, the sequences $\vec{x}, \vec{y}$ and $\vec{z}$ are sorted by networks of depth $\lceil\log(|\vec{x}|)\rceil$, $\lceil\log(|\vec{y}|)\rceil$ and $\lceil\log(|\vec{z}|)\rceil$, respectively.

In this section we present the *Asymmetric Tri-section technique* in which $|\vec{x}| = k$, $|\vec{y}| = n$ and $|\vec{z}| = n - k$, where k is an arbitrary number smaller than or equal to n. The technique is called "Asymmetric" in contrast to the "Symmetric" variant in which $|\vec{x}| = |\vec{z}|$.

The depth 1 network performing the Asymmetric Tri-section, with these parameters, is called $T^{k,n}$. Fig. 2 presents the network $T^{5,11}$. In this figure, a comparator is denoted as in Fig. 3. Namely, it contains two horizontal edges: a **Min** edge and a **Max** edge, connected by a diagonal line. The names of the edges entering this comparator are written on the diagonal line while the names of the edges coming out of it are written on the edges (see Fig. 3).

The general network, $T^{k,n}$, naturally follows the format of Fig. 2 and a formal definition is omitted. Note that the network $T^{0,n}$ is identical to the first stage of Batcher's Bitonic sorter [2]. Hence, in some sense, the Tri-section technique is a generalization of Batcher's technique.

Let $\bar{T}^{k,n}$ denote the mapping performed by the network $T^{k,n}$. That is, $\bar{T}^{k,n}(\langle \vec{a}, \vec{b}\rangle) = \langle \vec{x}, \vec{y}, \vec{z}\rangle$, where $\vec{x}, \vec{y}$ and $\vec{z}$ are the sequences generated by $T^{k,n}$ when it receives the input vector $\langle \vec{a}, \vec{b}\rangle$. To study $\bar{T}^{k,n}$ we name vectors of several types. A sequence $\vec{x}$ is *sorted* (*ascending*) if $x_i \le x_j$ whenever $i \le j$. Similarly, $\vec{x}$ is *descending* if $x_i \ge x_j$ whenever $i \le j$. A sequence is *ascending–descending* if it is a concatenation of an ascending sequence followed by a descending sequence. Similarly, a concatenation of a descending sequence followed by an ascending sequence is called *descending–ascending*. Note that either of the sequences may be empty; therefore, ascending sequences and descending sequences are both ascending–descending and descending–ascending. A sequence is *Bitonic*[4] if it is a rotation of an ascending–descending sequence. A comparator network is an *ascending–descending sorter* if it sorts all ascending–descending sequences. Similarly, we define *descending–ascending sorter* and *Bitonic sorter*.

A powerful tool for studying merging networks, similar to the 0–1 Principle, is the set of *Sandwich vectors*, presented in [4]. As demonstrated in the proof of Lemma 4, their simple and elegant form simplifies the analysis of merging networks

[4] This term was coined by Batcher [2] and we follow his terminology. We caution the reader that some authors use the term "Bitonic" with other meanings.

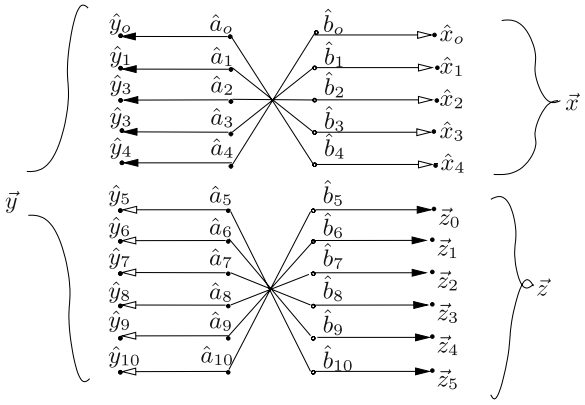

Fig. 2. The network $T^{5,11}$.

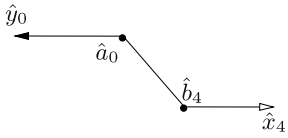

Fig. 3. A comparator receiving a_0 and b_4 and producing y_0 and x_4.

and leads to fewer special cases than the traditional 0–1 Principle. For the sake of sandwiches we assume that $\mathcal{K} = \mathbb{N}$. A *sandwich of width* $2n$ is a bisorted vector $\langle \vec{a}, \vec{b} \rangle$ in which every member of the interval $[0, 2n)$ appears exactly once and the range of the $\vec{a}$ sequence is an interval. The term "sandwich" follows from the fact that the vector can be sorted by inserting the sequence $\vec{a}$ consecutively in a certain place in the sequence $\vec{b}$. The sandwich technique is based on the following lemma:

Lemma 3 (*The Sandwich Lemma [4]*). *A network is a merging network if and only if it sorts all sandwiches.*

The following lemma is the keystone of the Asymmetric Tri-section technique.

Lemma 4. *Let $\langle \vec{a}, \vec{b} \rangle$ be a bisorted vector of width $2n$, let $k \in [0, n)$ and let $\langle \vec{x}, \vec{y}, \vec{z} \rangle = \bar{T}^{k,n}(\langle \vec{a}, \vec{b} \rangle)$. Then*

(1) $|\vec{x}| = k, \quad |\vec{y}| = n, \quad |\vec{z}| = n - k.$
(2) *Every key in $\vec{x}$ is smaller than or equal to any key in $\vec{y}$ and every key in $\vec{y}$ is smaller than or equal to any key in $\vec{z}$.*
(3) *$\vec{x}$ is ascending–descending, $\vec{z}$ is descending–ascending.*
(4) *$\vec{y}$ is Bitonic.*

Proof. Statement (1) follows directly from our construction. Statement (2) is straightforward and its proof omitted. The fact that $\vec{x}$ and $\vec{z}$ are Bitonic is proved by Batcher [2, Appendix B], in a slightly different context. In order to show that $\vec{x}$ is ascending–descending it suffices to show that the minimal key of $\vec{x}$ is in either the first position or the last position. This clearly follows from the fact that the key in question is either a_0 or b_0. The fact that $\vec{z}$ is descending–ascending is proved analogously and Statement (3) follows.

The hard part of this proof is showing Statement (4). To use the 0–1 Principle or the Sandwich Lemma we need the network in question to be a merging network. To this end, we extend the network $T^{k,n}$ into a network M as follows: The sequence $\vec{x}$ enters an arbitrary ascending–descending sorter, the sequence $\vec{y}$ enters an arbitrary Bitonic sorter and the sequence $\vec{z}$ enters an arbitrary descending–ascending sorter. We now prove that M is a merging network. This fact can be proved using (a variant of) the 0–1 Principle but this leads to many special cases which need to be verified. On the other hand, sandwiches lead to a proof having only two symmetric cases. Therefore, we next assume that the input $\langle \vec{a}, \vec{b} \rangle$ is a sandwich and show that in this case the sequence $\vec{y}$ is Bitonic.

Note that a sandwich vector $\langle \vec{a}, \vec{b} \rangle$ is determined by the key a_0. There are two (overlapping) cases; either $a_0 \leq k$ or $a_0 \geq k$. The two cases are similar and we focus on the first. Fig. 4 depicts the network $T^{5,11}$ processing the sandwich with $a_0 = 2$. The initial part of $\vec{y}$, having $k - a_0$ keys, comes from $\vec{b}$ in reverse order. Hence, this initial part of $\vec{y}$ is descending. The rest of $\vec{y}$ comes from $\vec{a}$ in the natural order; hence, this part is ascending. To summarize, in the case of $a_0 \leq k$, $\vec{y}$ is Bitonic. In the other case, where $a_0 \geq k$, the sequence $\vec{y}$ is ascending–descending; hence, $\vec{y}$ is Bitonic also in this case. This and the Sandwich Lemma imply that M is a merging network.

If we were just to prove that M is a merging network, then the proof would have ended here. However, our lemma is stronger — it says that $\vec{y}$ is always Bitonic, for any bisorted input. To prove that, assume for a contradiction that $\vec{y}$ is not Bitonic. By the following Lemma 6 (whose proof is not dependent on the current lemma), there exists a Bitonic sorter that does not sort $\vec{y}$. As we said, M processes $\vec{y}$ using an arbitrary Bitonic sorter. In particular, this Bitonic sorter could be the one that does not sort $\vec{y}$. This contradicts the fact that the entire network M is a merging network. ■

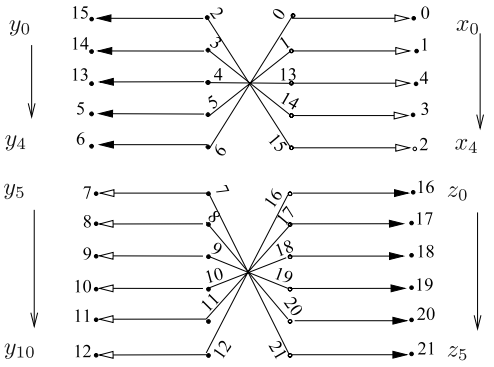

Fig. 4. The network $T^{5,11}$ processing the sandwich with $a_0 = 2$.

Our goal now is to show that for every non-Bitonic vector there exists a Bitonic sorter which does not sort it. To this end, we use the following result of [4]. For any 0–1 vector there is a network that identifies it in the following sense.

Lemma 5 (*The Identification Lemma, [4]*). *For every* 0–1 *vector* v *which is not constant, there is a network that sorts all the* 0–1 *vectors of the appropriate width, except* v.

Lemma 6. *For every non-Bitonic sequence there is a Bitonic sorter that does not sort it.*

Proof. We say that a vector $\vec{u}'$ is a *binary image* of a vector $\vec{u}$ if and only if there exists a monotonic function $f : \mathcal{K} \to \{0, 1\}$ such that $u'_i = f(u_i)$ for every i. It is not hard to see that a vector is Bitonic if and only if all its binary images are Bitonic. Let $\vec{v}$ be a non-Bitonic vector and let $\vec{v}'$ be a non-Bitonic binary image of $\vec{v}$.

Clearly, $\vec{v}'$ is not constant. By Lemma 5, there exists a network N of width $|\vec{v}|$ that sorts all binary vectors except $\vec{v}'$. By a straightforward 0–1 argument, a network sorts a vector if and only if it sorts all its binary images. Since all binary images of Bitonic vectors are Bitonic it follows that N is a Bitonic sorter. Since N does not sort $\vec{v}'$ it does not sort $\vec{v}$. ■

Returning to the Tri-section technique, recall that our goal is to construct a merging network of minimal depth in which the sequences $\vec{x}$, $\vec{y}$ and $\vec{z}$ are sorted by networks of depth $\lceil \log(|\vec{x}|) \rceil$, $\log(|\vec{y}|)$ and $\lceil \log(|\vec{z}|) \rceil$ respectively. To this end, we use pruning ([5,12], [8, Section 4.2.2]) to reduce the width of certain comparator networks. This technique is based on the concept of degenerate comparators, as defined in the introduction. Pruning, in the context of merging networks, is studied in [8] and can be applied as follows. Several consecutive input edges at the top of the input sequences are fed with the fictive values of $+\infty$ while the rest of the inputs are fed with real keys. Clearly, the paths of the fictive values are fixed — they do not depend on the values of the real keys. Any comparator that is on such a path is degenerate and can be removed without affecting the network's functionality. The resulting network, that processes no fictive values, is a merging network of a smaller width. Returning to our $\vec{x}$, $\vec{y}$ and $\vec{z}$, we first consider the case where n is a power of 2. In this case, the vector $\vec{y}$ is Bitonic and its width is a power of 2. Such a vector can be sorted by Batcher's Bitonic sorter [2], whose depth is $\log(n)$.

Concerning the sequence $\vec{x}$, recall that this sequence is not only Bitonic, but also ascending–descending. Such a sequence can be expanded into a Bitonic sequence of a desired width by adding fictive keys of value $-\infty$ at the beginning (or the end) of the sequence. This implies that any wide enough Bitonic sorter can be pruned into an ascending–descending sorter of a smaller given width. The depth of the resulting network is clearly not greater than the depth of the original one. Therefore, $\vec{x}$ can be sorted by a network of depth $\lceil \log(|\vec{x}|) \rceil$. By symmetry, the sequence $\vec{z}$ can be sorted in a similar manner.

Next consider the case where n is not a power of 2. Note that in this case $|\vec{y}|$ is not a power of 2. One may assume that a wider Bitonic sorter can be pruned into a Bitonic sorter of the desired width; however, as shown in [9], the minimal depth of a Bitonic sorter is not monotonic in the width of the input; therefore, such pruning is impossible.

The problem is solved as follows. Let $n' = 2^{\lceil \log(n) \rceil}$ be the first power of 2 following n. Let M' be the merging network of width $2n'$ generated by the asymmetric construction in which the k lowest outputs are accelerated to a depth of $\lceil \log(k) \rceil + 1$. The network M' is pruned into a width $2n$ merging network as discussed above. This results in a merging network N of width $2n$ and of depth $\lceil \log(2n) \rceil$ whose k lowest outputs are accelerated to a delay of $\lceil \log(k) \rceil + 1$ comparators.

Our construction possesses an additional important attribute which enables the concatenation of several such accelerating merging networks into an accelerating sorting network. This attribute relates to 'restricted reachability' as follows. We say that an edge e of a merging network *is reachable only from the k lowest (highest) inputs* if there is no path to e from an input edge which is not one the k lowest (highest) input edges of one of the input sequences. The construction of this section is summarized in the following lemma.

Lemma 7. *For every* $k < n$ *there is merging network of width* $2n$ *and of depth* $\lceil \log(2n) \rceil$ *in which each of the k lowest (highest) outputs is accelerated to a depth of* $\lceil \log(k) \rceil + 1$ *comparators and is reachable only from the k lowest (highest) inputs.*

4. The Symmetric Tri-section merging technique

Another Tri-section technique that accelerates certain outputs is presented in this section. As we said, the Tri-section technique separates, using a depth 1 network, a bisorted input into three sets, $\vec{x}$, $\vec{y}$ and $\vec{z}$ such that every key in one set is smaller than or equal to any key in the following set. We first consider the case where the width of the input, $2n$, and the number of accelerated outputs, k, are powers of 2. In this case the Symmetric Tri-section satisfies $|\vec{x}| = |\vec{z}| = k$. Hence, the network accelerates both the lowest k outputs and the highest k outputs to a delay of $\log(2k)$ comparators.

The symmetric technique is based on the Bitonic sorting technique of Nakatani et al. [11] which considers the keys to be arranged in a matrix. To this end, we denote a matrix of keys by $\vec{\vec{m}}$. Their technique is based on the following lemma.

Lemma 8 (*The Matrix Technique [11]*). *Let $\vec{b}$ be a Bitonic sequence of width $j \cdot k$ and $\vec{\vec{m}}$ be the $j \times k$ matrix having the $\vec{b}$ sequence in a row major fashion. Then:*

(1) *Every row in $\vec{\vec{m}}$ is a Bitonic sequence.*

(2) *Every column in $\vec{\vec{m}}$ is a Bitonic sequence. Let $\vec{\vec{m}}'$ be derived from $\vec{\vec{m}}$ by sorting each column separately. Then:*

(3) *Every row in $\vec{\vec{m}}'$ is a Bitonic sequence.*

(4) *Every element of every row in $\vec{\vec{m}}'$ is smaller than or equal to any element of the next row.*

The matrix technique of Nakatani et al. for sorting Bitonic sequences of length $j \times k$ is composed of three stages:

Stage 1: Arrange the Bitonic sequence in a $j \times k$ matrix in a row major fashion.
Stage 2: Independently, sort every column using a Bitonic sorter.
Stage 3: Independently, sort every row using a Bitonic sorter.

Following those stages the resulting matrix is sorted in a row major fashion.

Note that the matrix technique does not require that j and k be powers of 2; however, we use it only under this restriction. Assume that the Bitonic sorters used in stage (2) and stage (3) are of minimal depth. The depth of the entire network is $\log(j) + \log(k)$ which is minimal. As shown in [8], for every n, a power of 2, there is a unique n-key Bitonic sorter of minimal depth. This implies that, when the width is a power of 2, the network of Nakatani et al. is identical to Batcher's [2] Bitonic sorter. Yet, even in this case, Nakatani's technique sheds a new light on Batcher's Bitonic sorter.

Note that the parameter k in the technique of Nakatani et al. and the parameter k of our technique refer to the same number; namely, using a $j \times k$ matrix as per Lemma 8, we accelerate the highest k outputs and the lowest k outputs. Since we are only interested in merging and not in Bitonic sorting, we can perform stage (2) in a special manner. When the bisorted input is turned into a Bitonic sequence and arranged in the above matrix, every column in the matrix $\vec{\vec{m}}$ is not only Bitonic, but also in fact bisorted; hence, it can be sorted by any merging network. There are many merging networks [8, Section 6] of minimal depth that produce the lowest key and highest key after a delay of a single comparator (e.g., Batcher's [2] odd–even merging network); therefore, stage (2) can be performed by k such merging networks, working in parallel. Stage (3) can be performed by j Batcher Bitonic sorters, working in parallel. The depth of these Bitonic sorters is minimal $-\log(k)$; therefore, this construction accelerates the lowest k outputs and highest k outputs to a delay of $\log(k) + 1$ comparators.

Let $\tilde{M}$ be the network performing Stage (2). By Statement (4) of Lemma 8, $\tilde{M}$ Tri-sects the keys into three sets: the first row, the last row and all the rest. Moreover, the very same Tri-section is performed by a subnetwork of $\tilde{M}$ of depth 1; hence this technique is a Tri-section technique. Our construction not only accelerates the required inputs but also has a restricted reachability as summarized in the following lemma.

Lemma 9. *For any $k < n$, both powers of 2, there is merging network of width $2n$ and of depth $\log(2n)$ in which:*

- *Each of the k lowest outputs is accelerated to a delay of $\log(2k)$ comparators and is reachable only from the lowest k inputs.*
- *Each of the k highest outputs is accelerated to a delay of $\log(2k)$ comparators and is reachable only from the highest k inputs.*

Next consider the case where k is not a power of 2. As in the previous section, instead of accelerating k outputs we accelerate $k' = 2^{\lceil \log k \rceil}$ outputs. Note that in this construction, each accelerated output is reachable from $k' = 2^{\lceil \log k \rceil}$ (rather than k) extreme inputs.

Finally, consider the case where the network's width, $2n$, is not a power of 2. In this case, we do not know how to accelerate both the highest k and the lowest k keys, simultaneously; in fact, we do not know if such acceleration is possible. We do know how to accelerate either the smallest k outputs or the highest k outputs. This is accomplished by pruning a network whose width is a power of 2. The following lemma (similar to Lemma 7) summarizes this case.

Lemma 10. *For any $k < n$ there is merging network of width $2n$ and of depth $\lceil \log(2n) \rceil$ in which each of the k lowest (highest) outputs is accelerated to a delay of $\lceil \log k \rceil + 1$ comparators and is reachable only from the lowest (highest) $2^{\lceil \log k \rceil}$ inputs.*

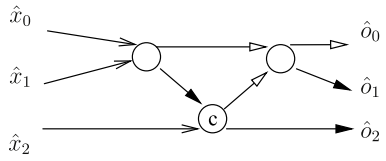

Fig. 5. An ascending–descending sorter.

5. A counterexample

As shown in [8], all published merging networks (whose width is a power of 2) are Batcher merging networks. Namely, they are constructed by a straightforward generalization of Batcher's odd–even technique. The depth of all these merging networks is minimal. This raises the following question:

Question 1. *Are the Batcher merging networks the only merging networks with the following properties:*

1. *Their width,* $2n$*, is a power of 2.*
2. *Their depth is minimal* — $\log(2n)$.
3. *They have no degenerate comparators.*

The answer to this question is no. The Tri-section technique can generate a counterexample based on the fact that when $|\vec{x}|$ is small w.r.t. n, the sequence $\vec{x}$ can be sorted in an arbitrary manner (using a network of excessive depth) while maintaining the minimal depth of the entire merging network. We next present such a network for any $n \geq 8$, a power of 2.

Our construction starts with the network $T^{3,n}$ that produces the three sequences $\vec{x}$, $\vec{y}$ and $\vec{z}$. The sequence $\vec{x}$ is sorted by the network depicted in Fig. 5 which has no degenerate comparators. The sequences $\vec{y}$ and $\vec{z}$ are sorted by any minimal depth network as per Section 3. The resulting merging network, M, satisfies the three conditions of Question 1. (If M has degenerate comparators, they should be removed.) The network M is not a Batcher merging network since it has a comparator, c, with the following property. Of the two edges exiting c, one is the output edge $\hat{o}_2$ and the other is not an output edge. This is never the case in a Batcher merging network.

The above construction can be extended to yield a merging network of minimal depth which does not follow the "divide and conquer" paradigm. Let $k = |\vec{x}|$ be large enough and still much smaller than n. Then the sequence $\vec{x}$ can be sorted using a network which is clearly not of the above paradigm. Two such examples are Knuth's bubble-sort network and Knuth's odd–even transposition sort [6, pp 223, 241]. This construction may produce degenerate comparators that can be removed without affecting the network's functionality. This implies the existence of a minimal depth merging network that has no degenerate comparators and has an arbitrary large subnetwork lacking any recursive structure.

6. Accelerating sorting networks

Building on the merging networks introduced in previous section, we now utilize the classical merge-sort technique to construct a sorting networks that accelerates certain outputs. The merge-sort technique divides the input keys into two parts whose size differ by at most one and recursively sorts each part; it then merges the two resulting sorted sequences into a single sorted sequence. When the number of keys is not a power of 2, the merge-sort technique requires merging sequences of different size. So far this paper focused on merging two sequences of equal size. However, by pruning, our constructions are applicable to merging of sequences of different size.

This section presents a merge-sort network of width n and of depth $\frac{\lceil\log(n)\rceil\cdot\lceil\log(2n)\rceil}{2}$ which accelerates the k lowest outputs (or k highest outputs) to a delay of less then $\lceil\log(n)\rceil\cdot\lceil\log(2k)\rceil$. This depth is the same as the depth of Batcher's merge-sort networks [2]. We, therefore, refer to this depth, $\frac{\lceil\log(n)\rceil\cdot\lceil\log(2n)\rceil}{2}$, as the *Batcher depth*. In theory, due to the AKS construction [1], there are sorting networks whose depth is much lower than the depth of merge-sort networks. However, this holds only for impractically large n. The merge-sort networks of Batcher, invented in 1968, are still the best practical sorting networks [6, pp. 228].

A natural question in this context is the following one: Is the Batcher depth the minimal depth of a merge-sort network? It seems that the answer to this question is positive but we can prove it only for restricted cases. The naive approach to answer this question goes as follows. The depth of a network that merges a sorted sequence of length j with a sorted sequence of length k is at least $\lceil\log(j+k)\rceil$ [6, pp. 243]. Summing these depths for a merge-sort network yields the Batcher depth. This argumentation does not work due to the following difficulty. The depth of the stages of a merge-sort network does not necessarily sum up. Namely, the depth of a concatenation of several networks may be smaller than the sum of depths of the networks.

We can prove that the Batcher depth is minimal for merge-sort networks only for the special case where n is a power of 2. In this case, the last stage of a merge-sort network is a network that merges two sequences of the same length, $n/2$, which is a power of 2. By a simple reachability argument, every input of this stage has a path of depth $\lceil\log(n)\rceil$ starting at this input. By induction, the depth of the entire merge-sort network is at least a Batcher depth.

We assume, without loss of generality, that k is a power of 2. By pruning, we may also assume that n is a power of 2. Our construction is composed of a sorting stage followed by $\log(n) - \log(k)$ merging stages. In the sorting stage the n input keys are divided into sets of k keys each, and each of these sets is sorted separately using any sorting network of a Batcher depth. We now follow the merge-sort method. Namely, in each of the merging stages, all the sorted sequences produced in the previous stage are paired and each pair is merged into a single sorted sequence. This merge is performed by a merging network, as per Lemma 7 or Lemma 10, that accelerates its k lowest outputs to a delay of $\log(2k)$ comparators and, moreover, it possesses the restricted reachability property.

Consider the delay of the lowest k outputs. This delay is composed of $\frac{\log k \cdot \log 2k}{2}$ comparators in the sorting stage and $\log 2k$ comparators in each of the $\log n - \log k$ merging stages. Due to the restricted reachability property, these delays are added up; that is, in the entire network, the delay of the lowest k outputs is at most $\frac{\log k \cdot \log 2k}{2} + (\log n - \log k) \cdot \log 2k$. Clearly, the depth of the entire sorting network is a Batcher depth. Our construction is summarized in the following lemma.

Theorem 11. *For every $0 < k < n$, there is a sorting network of width n and of Batcher depth that accelerates all the lowest k (or highest k) outputs to a delay of $\frac{\log k \cdot \log 2k}{2} + (\log n - \log k) \cdot \log 2k$.*

In the special case where n is a power of 2, we may use Lemma 9 to achieve the above acceleration both for the highest k keys and for the lowest k keys, simultaneously.

References

[1] M. Ajtai, J. Komlós, Szemerédi, Sorting in c log n parallel steps, Combinatorica (1) (1983) 1–19.
[2] K.E. Batcher, Sorting networks and their applications, Proc. AFIPS Spring Joint Computer Conf., 1968, vol. 32, pp. 307–314.
[3] K.E. Batcher, K.J. Liszka, A modulo merge sorting network, in: Fourth Symposium on the Frontiers of Massively Parallel Computation, 1992, pp. 164–169.
[4] G. Even, T. Levy, A. Litman, Optimal conclusive sets for comparator networks, Structural Information and Communication Complexity, 2007, pp. 304-317, Theoretical Computer Science, in press (doi:10.1016/j.tcs.2008.12.021).
[5] M.W. Green, Cellular Logic in Memory Arrays, Final Report, SRI Project 5509, Part 1, Stanford Research Institute, 1970, pp. 49–71.
[6] D.E. Knuth, The Art of Computer Programming vol. 3: Sorting and Searching, second edition, Addison-Wesley, 1998.
[7] F.T. Leighton, Introduction to Parallel Algorithms and Architectures, Morgan Kaufmann, 1991.
[8] T. Levy, A. Litman, On merging networks., Technical Report CS-2007-16. Technion, Department of Computer Science, 2007.
[9] T. Levy, A. Litman, Lower bounds on Bitonic sorting networks, Technical Report CS-2009-07. Technion, Department of Computer Science.
[10] T. Levy, A. Litman, The strongest model of computation obeying 0-1 Principles, Technical Report CS-2007-17. Technion, Department of Computer Science, 2007.
[11] T. Nakatani, S.T. Huang, B.W. Arden, S.K. Tripathi, k-way Bitonic sort, IEEE Trans. Comput. 38 (1989) 283–288.
[12] C.C. Yao, Bounds on selection networks, SIAM J. Comput. 9 (1980) 566–582.