

Improving Bitonic Sorting by Wire Elimination

Moritz Mühlenthaler and Rolf Wanka, Department of Computer Science, University of Erlangen-Nuremberg, Germany. {moritz.muehlenthaler, rwanka}@informatik.uni-erlangen.de

Abstract

We introduce a technique called *wire elimination* by which it is possible to remove wires and comparators from (n, m) -merging and n -sorting circuits such that the resulting circuits are (n', m') -merging and n' -sorting circuits, resp., with $n' < n$, $m' < m$. By neatly choosing the wires to be removed, it is possible to obtain for n' and m' new circuits that have size less than circuits previously designed for n' and m' . We demonstrate this approach by eliminating from the classical Bitonic $(2n, 2n)$ -merging circuit $2n$ wires such that an (n, n) -merging circuit is obtained which has $\frac{1}{2}n$ comparators less than the classical Bitonic (n, n) -merge circuit, but still the same depth. Using the usual sorting by merging technique, we get a variant of Bitonic sort which saves $\frac{1}{4}n(\log n - 1)$ comparators compared to the classical variant.

1 Introduction

Background. Sorting n keys is one of the most thoroughly investigated problems in computer science. In the area of parallel computing, sorting is a classical topic as well. Its roots can be traced back to the 1950s [6, Sec. 5.3.4, p. 225]. Parallel sorting is one of the basic subroutines which appears as central building block in many applications like the computation of convex hulls, image processing methods, and parallel databases.

One approach to parallel sorting is to design special circuits for sorting which consist exclusively of modules called *comparators* realizing atomic compare-exchange operations (for detailed definitions, see Section 2).

Batcher [2] published in 1968 two circuits called the *Bitonic* sorting circuit and the *Odd-Even Merge* sorting circuit (OEMS). For $n = 2^k$, both algorithms perform $\frac{1}{2} \log n \cdot (\log n + 1)$ parallel steps. The Bitonic sorter consists of $\frac{1}{4}n \log n (\log n + 1)$ comparators, the Odd-Even Merge sorter of $\frac{1}{4}n \log n (\log n - 1) + n - 1$ comparators. So OEMS has $\frac{1}{2}n(\log n - 2) + 1$ comparators less than the Bitonic sorter. The size of OEMS is up to now the best known for a general construction of sorting circuits. Only by extensive search for individual n , sorting circuits of smaller size have been found (see [6, p. 229]).

Bitonic sort is quite popular as its structure is well suited to be implemented on processor networks like k -dimensional hypercubes and 2-dimensional meshes. In comparison, the structure of the Odd-Even Merge sorter is quite irregular which impedes direct implementation on the mentioned networks.

If the number n of keys is larger than the number p of processors of a parallel processor network, then a comparator can be replaced by the so-called *split-and-merge* operation, where the “small” half of the keys is treated like the minimum key in the comparator case, and the “large” half correspondingly [6, p. 241]. Due to the small number of split-and-merge operations, in this case, implementations of Odd-Even Merge sort on SIMD parallel machines

are faster than any other method [4]. Also on MIMD parallel machines, OEMS is competitive to non-comparator-based methods like Sample Sort for a substantial range for n/p [10].

The theoretical lower bound on comparator-based parallel sorting is $\Omega(\log n)$. The only known sorting circuit matching this bound is the AKS sorting circuit [1, 9]. However, the constant factor hidden in the O -notation is astronomically large, so the AKS circuit is mainly of theoretical relevance.

Unfortunately, even for smaller circuits it is often far from trivial to *prove* that they sort all inputs. In this paper, we investigate a general transformation of sorting circuits which may save comparators and preserves the sorting property of the original circuit, so the new circuit is correct by construction.

Previous work. There has been some work on modifying sorting circuits in order to preserve its properties and guarantee additional properties. E.g., Knuth shows [6, p. 238] how to transform any sorting circuit consisting of so-called standard and nonstandard comparators into a version that consists only of standard comparators (see Fact 1 below).

Kutyłowski *et al.* [7] present the periodification scheme that transforms any sorting circuit into a periodic sorting circuit that consists of identical blocks of depth 5 and has almost the same overall depth.

Leighton [8] shows how any n -sorting circuit can be transformed into a parallel constant-degree processor network of n processors which can sort in parallel time of the circuit’s depth.

Closest to our work, Hong and Sedgewick [5] use a simple variant of what we call wire elimination for reducing the depth of a particular class of odd-even mergers.

New results. In this paper, we introduce a technique we call *wire elimination* by which it is possible to modify an n -sorting circuit or an (n, m) -merging circuit such that an

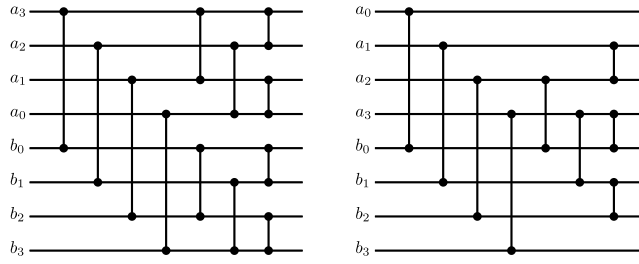


Figure 1: The Bitonic (4,4)-merger $BM_{4,4}$ and the Odd-Even (4,4)-merger $OEM_{4,4}$, and the corresponding input orders

n' -sorting circuit is obtained and an (n', m') -merging circuit, resp., for $n' < n$ and $m' < m$. The basic idea is that if S_n is an n -sorter, we may fix an input wire i , put, e.g., the key $-\infty$ to i and follow the path of $-\infty$ through the circuit assuming larger keys on the other wires. As $-\infty$ loses all comparisons, its path can be removed from the circuit, including all incident comparators. The residual circuit must be an $(n-1)$ -sorting circuit, with possibly less comparators or depth than other known constructions for $n-1$. No analysis of the structure of the $(n-1)$ -sorter or brute-force approach is needed for a correctness proof, it is correct by construction. Similar arguments can be applied to merging circuits.

We demonstrate the wire elimination approach by applying it to the Bitonic sorter. First, we construct from the Bitonic $(2n, 2n)$ -merging circuit an (n, n) -merging circuit which saves $\frac{1}{2}n$ comparators in comparison to the classical Bitonic (n, n) -merger. Then, the classical sorting by merging approach is applied to the new variant of the Bitonic (n, n) -merger such that we obtain an n -sorting circuit which saves $\frac{1}{4}n(\log n - 1)$ comparators in comparison to the classic Bitonic n -sorter. So we have a variant of Bitonic sort which is, with respect to the size, closer to OEMS than to the classical Bitonic n -sorter.

Organization of paper. In the next section, we present the necessary definitions and facts from the area of parallel sorting and merging circuits. Section 3 presents the wire elimination, which is applied to the Bitonic merge and sort in Section 4.

2 Preliminaries

A *comparator* is a module with two inputs x_1 and x_2 , and two outputs y_1 and y_2 . Two arbitrary keys a and b enter the comparator, and, as the output of the comparator, y_1 receives the smaller key, i. e., $y_1 = \min\{a, b\}$, and y_2 the larger key, i. e., $y_2 = \max\{a, b\}$.

A comparator is graphically represented as an arrow. The arrow always shows the direction of the maximum key. A comparator n -circuit consists of n wires that go from left to right. The wires are numbered from 1 through n , and they are connected by comparators. The comparator $[i : j]$ connects the wires i and j , and the maximum of the inputs is placed on wire j . If $i < j$, the comparator $[i : j]$ is called *standard* comparator. In this case, we omit the arrow in the representation.

Comparators can be combined to a *parallel step* or *stage* if there are no dependencies between their inputs and outputs. The number of parallel steps is the *depth* of the circuit, the number of comparators is its *size*. For example, see **Figure 1**, where circuits of depth 3 and size 12 and 9, resp., are presented.

If a comparator n -circuit sorts any input sequence of length n , it is called *n -sorter*.

Let the wires be partitioned into two groups of size n and m , and let there be an order on the wires of each group. If we can input to the groups any two sorted sequences according to the order, and the output leaves the circuit always sorted, the circuit is called *(n, m) -merger*.

Fact 1 ([6, p. 238]). *Any sorting circuit consisting of standard and nonstandard comparators can be transformed into a sorting circuit consisting only of standard comparators having the same depth and size.*

Similarly, any (n, m) -merging circuit can be transformed into an (n, m) -merging circuit consisting only of standard comparators having the same depth and size, but possibly with new groups (and order).

Fact 2 ([2]). *Let $n = 2^k$.*

- (a) *The Bitonic (n, n) -merger $BM_{n,n}$ has depth $\log n + 1$ and size $n(\log n + 1)$. The Bitonic n -sorter BS_n has depth $\frac{1}{2} \log n(\log n + 1)$ and size $\frac{1}{4}n \log n(\log n + 1)$.*
- (b) *The Odd-Even (n, n) -merger $OEM_{n,n}$ has depth $\log n + 1$ and size $n \log n + 1$. The Odd-Even Merge n -sorter $OEMS_n$ has depth $\frac{1}{2} \log n(\log n + 1)$ and size $\frac{1}{4}n \log n \cdot (\log n - 1) + n - 1$.*

So the Odd-Even Merge sorter has $\frac{1}{2}n \log n - n + 1$ comparators less than the Bitonic sorter.

Figure 1 shows the Bitonic (4,4)-merger and the Odd-Even (4,4)-merger together with the groups of wires and the order within these groups.

Knuth describes variants of $BM_{m,n}$ [6, p. 230ff.] and of $OEM_{m,n}$ [6, p. 223ff.] that work for arbitrary n and m .

In this paper, we construct a variant of the Bitonic sorter that halves the difference in the size of the two circuits.

3 Wire Elimination

Wire elimination is a general transformation operation on comparator circuits which preserves the sorting property of the input circuit. The general idea is to reduce the number

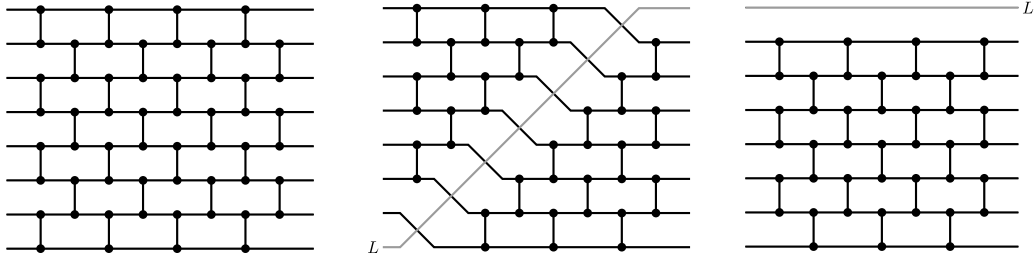


Figure 2: Elimination of one wire from Odd-Even Transposition Sort on 8 wires. Note that 7 comparators are removed, stages are merged and that the residual circuit has one stage less.

of wires by fixing parts of the input sequence with nonstandard symbols $\{L, R\}$, where L is smaller and R is greater than any standard symbol. L may be interpreted as $-\infty$, and R as ∞ . In the following, we call regular keys standard symbols as their actual values are irrelevant for the elimination process. Since outcomes of all comparisons between standard and nonstandard symbols are known in advance the actions of the respective comparators (“swap symbols” or “pass through”) can be hardwired into the wire structure of the circuit, and the comparators can be omitted. As a result, the circuit is divided into two independent parts: one part dealing only with standard symbols and another part dealing only with nonstandard symbols. The latter part can be removed from the circuit, leaving us with a comparator circuit with fewer inputs. For example, see **Figure 2**.

Although wire elimination and related techniques have occurred in the parallel sorting literature, it has not received much attention as a general construction technique for sorting and merging circuits. An idea related to wire elimination has for instance been used for adapting fixed-width sorting circuits, implemented in hardware, to sort sequences of different lengths. To ensure that the size of the input sequence fits to the size of the sorter, a sequence is padded with copies of the largest or the smallest symbol. After sorting the extended sequence, the additional symbols are trimmed from the output to get a sorted sequence of the original length. For this technique only standard symbols are required. If wires are eliminated explicitly however, nonstandard symbols need to be introduced so the interaction between the standard symbols and the additional symbols is uniquely determined.

In the work of Hong and Sedgewick [5] explicit wire elimination has been used for creating merging circuits with fewer parallel steps than certain Odd-Even mergers. Their construction of a (k, l) -merger works as follows: From an Odd-Even (m, m) -merger $\text{OEM}_{m,m}$ with $m = 2^{\lceil \log_2(k+l) \rceil - 1}$ inputs the first $m - k$ wires are eliminated using L symbols and the last $m - l$ wires are eliminated using R symbols. Hong and Sedgewick show that if $\lceil \log_2(k+l) \rceil < \lceil \log_2 \max(k, l) \rceil + 1$ then the first parallel step is eliminated entirely from the circuit and the resulting (k, l) -merger has one parallel step less than $\text{OEM}_{k,l}$. In this construction, swaps of standard and nonstandard symbols are carefully avoided since such swaps result in twisted wires which complicate the circuit layout and make the resulting circuits difficult to analyze.

Algorithm WIREELIM shows how to implement wire

Algorithm 1: Wire elimination on n wires

Algorithm: WIREELIM

input : L : list of wire indices of the L symbols

input : R : list of wire indices of the R symbols

in/out : N : list of comparators $[i_1 : j_1], \dots, [i_r : j_r]$

output: Π : permutation of inputs

foreach $c = 1, 2, \dots, r$ **do**

if $i_c \in L \sqcup R \vee j_c \in L \sqcup R$ **then**

if $i_c \in R \vee j_c \in L$ **then**

$N[c] \leftarrow (i_c, j_c)$ (*)

 replace $[i_c : j_c]$ with transposition (i_c, j_c)

else

 remove $N[c]$

end

 update indices in L and R

end

end

$\Pi \leftarrow id_n$

foreach $c = r, r-1, \dots, 1$ **do**

if $N[c]$ is a comparator **then**

$i_c \leftarrow \Pi(i_c)$

$j_c \leftarrow \Pi(j_c)$

else

$\Pi \leftarrow (i_c, j_c) \circ \Pi$

end

end

remove wires with indices in $L \sqcup R$ and update Π accordingly

return N, Π

elimination as a general construction on comparator circuits. The input data are two lists of wire indices, one for L and one for R symbols (all indices must be distinct), and a comparator circuit N , which is represented as a list of comparators. The algorithm consists of a forward and a backward pass over N . In the first pass, the outcomes of all comparisons involving at least one nonstandard symbol are hardwired into the circuit structure. Suppose a comparator $[i : j]$ processes symbols a and b , with at least one of them nonstandard. Then at (*), the comparator is replaced by a transposition (i, j) if it swaps a and b . Otherwise the comparator is removed from the circuit entirely. Without loss of generality, we assume in WIREELIM that whenever two identical nonstandard symbols are compared, they are

swapped. In the second pass the transpositions are removed from the circuit, so we obtain a circuit represented as a list of comparators. The circuit is traversed back to front. The permutation π maps the current wire indices to outputs. Since the order of outputs is fixed, we start with $\pi = \text{id}_n$ and whenever a transposition is encountered, π is updated and the transposition is removed. For each comparator c , the wire indices i_c and j_c are set to the indices of the outputs the wires are connected to, namely $\pi(i_c)$ and $\pi(j_c)$. None of the modifications of the input circuit by WIREELIM affects the relative order of the standard symbols in the output sequence. The permutation π returned by the algorithm maps the wire indices of the original circuit to the modified circuit. The time complexity of the algorithm is $O(c)$, where c is the number of comparators.

Figure 2 shows a nice application of wire elimination to Odd-Even Transposition Sort, an n -sorter of depth n (see [6, p. 240]). Because of its simple structure – it only consists of so-called *primitive* comparators, i.e., comparators of the form $[i : i + 1]$ – wires are not twisted in this case. We see that the resulting residual 7-circuit must be a sorting circuit because the original circuit is an 8-sorter.

This observation holds in general.

Theorem 1. *Let N be an n -sorting circuit, and let L and R arbitrary disjoint sets of wire numbers. Then the result of Algorithm WIREELIM describes a sorting circuit with $n - |L| - |R|$ wires.*

Proof. As the original circuit is a sorter, the residual circuit still has to sort any sequence of regular keys because L and R are considered less and larger, resp., than any key in the sequence of regular keys. \square

Eliminating wires from merging circuits does not necessarily preserve their merging property. We now establish a condition under which the merging property of the input circuit is preserved. Let $a_0 \leq a_1 \leq \dots \leq a_{k-1}$ be k items processed by some comparator circuit. We say that the nondecreasing order of the items is respected when fixing items with nonstandard symbols, if (1) no standard or R symbol precedes an L symbol and (2) no standard or L symbol succeeds an R symbol.

Theorem 2. *Let M be a (k, l) -merging circuit merging items $a_0 \leq a_1 \leq \dots \leq a_{k-1}$ and $b_0 \leq b_1 \leq \dots \leq b_{l-1}$. Further, let E_M be a wire elimination operation fixing r items of a_0, \dots, a_{k-1} and s items of b_0, \dots, b_{l-1} with nonstandard symbols such that the nondecreasing order of items a_0, \dots, a_{k-1} and b_0, \dots, b_{l-1} is respected. Then the result of E_M is a $(k - r, l - s)$ -merging circuit.*

Proof. Analogously to the proof of Theorem 1. \square

Note the following: As the output wires define a sorted sequence the wire elimination process changes the actual order on the groups a and b .

Wire elimination can be used in two ways: On the one hand, it can be applied to a family of n -circuits of depth $T(n)$ and size $S(n)$ in such a way that certain, appropriately chosen sets of wires are eliminated so that a new family of

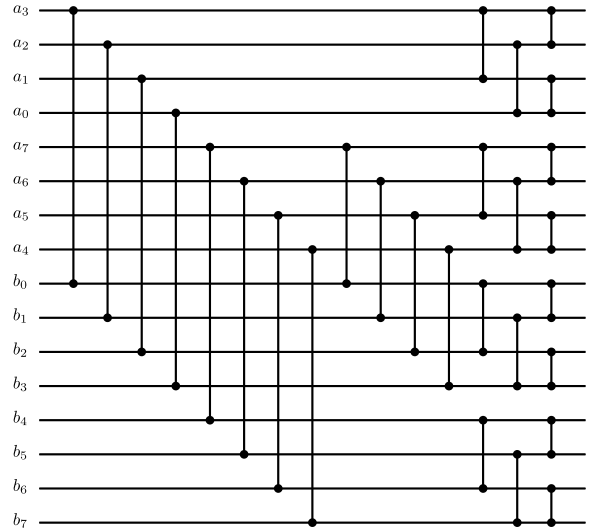


Figure 3: The $(8, 8)$ -improved Bitonic merger M'_{16}

n -circuits is found with depth $T'(n) < T(n)$ or size $S'(n) < S(n)$. In this paper, we construct from the family of Bitonic mergers $\text{BM}_{n/2, n/2}$ of size $\frac{1}{2}n \log n$ a family of $(n/2, n/2)$ -mergers of size $\frac{1}{2}n(\log n - \frac{1}{2})$ saving $\frac{1}{4}n$ comparators.

On the other hand, for fixed n and existing n -circuits, sets L and R of wires can be systematically tested by a computer program whether WIREELMIM with L and R leads to an n' -circuit with depth or size better than circuits previously known for n' .

4 Comparator Saving Variant of Bitonic Sort

The depth of Bitonic mergers is optimal [6, p. 231]. But the size of Bitonic mergers is larger than the size of Odd-Even Merging circuits (Fact 2). In this section, we provide a wire elimination-based construction which improves the size of the Bitonic merging circuit leaving the depth unchanged.

4.1 Improved Bitonic Merge Construction

First we describe the structure of the new $(n/2, n/2)$ -merger M'_n and the corresponding input order. Then we show that it can be obtained by wire elimination from the Bitonic $(n/2, n/2)$ -merger $\text{BM}_{n/2, n/2}$ which by Theorem 2 already proves its correctness.

M'_2 is a single comparator. For $n = 2^k$, $k > 1$, M'_n is defined as follows:

(1) Start with the Bitonic $(n/2, n/2)$ -merger $\text{BM}_{n/2, n/2}$ with $n = 2^m$ wires ($m \geq 2$).

(2) With $d = n/4$, replace the $n/2$ comparators in the second stage of the circuit with comparators $[d : 2d]$, $[d + 1 : 2d + 1]$, \dots , $[2d - 1, 3d - 1]$.

The items $a_0 \leq \dots \leq a_{n/2-1}$ and $b_0 \leq \dots \leq b_{n/2-1}$ are expected to arrive in the order: $a_{n/4-1}, \dots, a_0, a_{n/2-1}, \dots, a_{n/4}, b_0, \dots, b_{n/2-1}$.

Figure 3 shows the $(8, 8)$ -improved Bitonic merger M'_{16} and the order required for the inputs.

Theorem 3. Let M'_n be the circuit with $n = 2^k$ wires constructed in the above way. Then M'_n has the following properties:

1. M'_n is an $(n/2, n/2)$ -merging circuit.
2. M'_n has depth $T(n) = \log n$.
3. The size of M'_2 is $C(2) = 1$. For $k \geq 2$, M'_n has size $C(n) = \frac{1}{2}n(\log n - \frac{1}{2})$.

Note that M'_n has $\frac{1}{4}n$ less comparators than $BM_{n/2, n/2}$.

Proof. We first show that M'_n can be obtained by wire elimination from $BM_{n, n}$. Let $BM_{n, n}$ be the (n, n) -Bitonic merger merging items $a_0 \leq \dots \leq a_{n-1}$ and $b_0 \leq \dots \leq b_{n-1}$. We divide these items into eight groups a, \dots, h of $n/4$ consecutive items each, and eliminate wires from $BM_{n, n}$ such that L symbols are assigned to all items from groups a, b and e , and R symbols are assigned to all items in group h .

Figure 4 shows the groups a, \dots, h in the first three merging tableaux (for a description, see [5]) of $BM_{n, n}$. Blocks marked with a $*$ -symbol have their items arranged in reversed order. After the third merging tableau, all nonstandard symbols are already in the correct groups and hence need not be considered further. Since all standard symbols are now in blocks c^*, d^*, f and g the fourth merging tableau is equivalent to the third merging tableau of the (n, n) -Bitonic merger. In the first two merging tableaux, shown in Figures 4a and b only the comparisons $[c^* : f]$ and $[d^* : g]$ are not redundant. All comparisons between items in these blocks are independent, so these $n/2$ comparisons constitute the first stage of a circuit we call N' . Since we have transpositions (c^*, a^*) in the second merging tableau and no further transpositions in all following steps, removing the transpositions will result in relabeling comparisons $[c^* : f]$ to $[a^* : f]$. So indeed, the first stages of N' and M'_n are equivalent. The second stage of N' consists of comparisons $[d^* : f]$ (shown in the third merging tableau) and is equivalent to the second stage of M'_n . Since only groups c^*, d^*, f and g need to be considered after the second stage, the remaining stages of N' are equivalent to stages $3, \dots, \log n$ of a $(n/2, n/2)$ -Bitonic merger, and hence $M'_n = N'$. **Figure 5** shows the construction of M'_n by wire elimination for $n = 8$.

It remains to be shown that M'_n is indeed an $(n/2, n/2)$ -merging circuit expecting items in the specified order. Since L symbols are assigned to items in groups a, b and e , no standard or R symbols precede any L symbols. Similarly no standard or L symbols succeed the R symbols in group h . By Theorem 2, M'_n is an $(n/2, n/2)$ -merging circuit. Since no transpositions relevant for the relative order of the standard symbols occur after merging tableau 4c, the expected order of items can be derived from the order of the groups c^*, d^*, f and g (and the arrangement of items within these groups) in merging tableau 4c. So the expected relative order of items is

$$c_{n/4-1}, \dots, c_0, d_{n/4-1}, \dots, d_0, f_0, \dots, f_{n/4-1}, g_0, \dots, g_{n/4-1}.$$

Since we have $\forall r \in c^*, s \in d^* : r \leq s$ and $\forall r \in f, s \in g : r \leq s$, the expected order of items in M'_n matches the order specified in the construction.

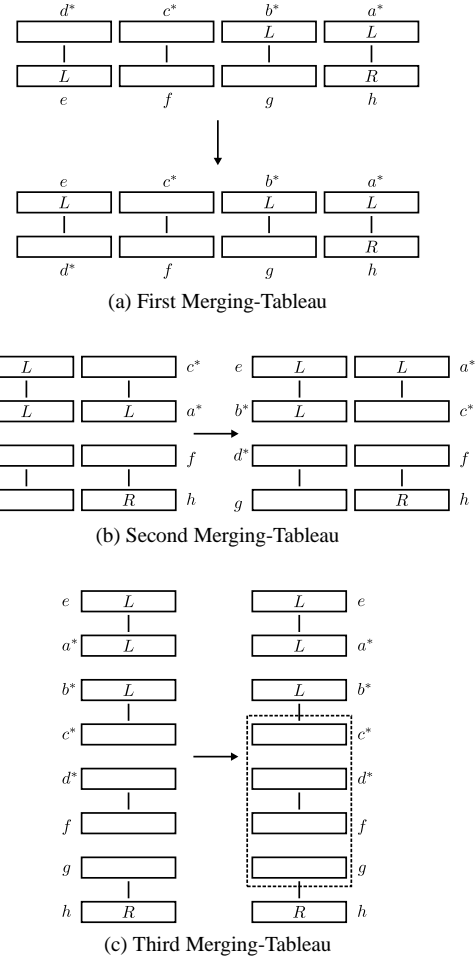


Figure 4: The first three merging tableaux of the Bitonic (n, n) -merging circuit. Each block contains $n/4$ symbols.

Claims 2 and 3 of the theorem follow immediately from the specification of the construction of M'_n . \square

The construction given in this section describes a class of improved Bitonic mergers with 2^m inputs for $m \geq 2$. We note that this construction can be generalized in the following way: Let M be an (n, n) -improved Bitonic merger ($n = 2^m, m \geq 2$). Then eliminating wires $0, \dots, n/4 - 1$ and $n, \dots, 5n/4$ with L symbols and wires $7n/4, \dots, 2n - 1$ with R symbols yields an $(n/2, n/2)$ -improved Bitonic merging circuit. This means, by eliminating wires from an (n, n) -improved Bitonic merger one by one, we can “interpolate” between an (n, n) - and an $(n/2, n/2)$ -improved Bitonic merger.

The merging circuit we investigated here is not the only one one can obtain by wire elimination. **Figure 6** presents four more variants of Bitonic $(4, 4)$ -mergers obtained from the classical Bitonic $(8, 8)$ -merger, and the corresponding input orders.

4.2 Improved Bitonic Sort Construction

Similar to the construction of Bitonic and Odd-Even sorters, the improved Bitonic merging circuit M'_n analyzed in the previous section can be used as a building block in a *divide-*

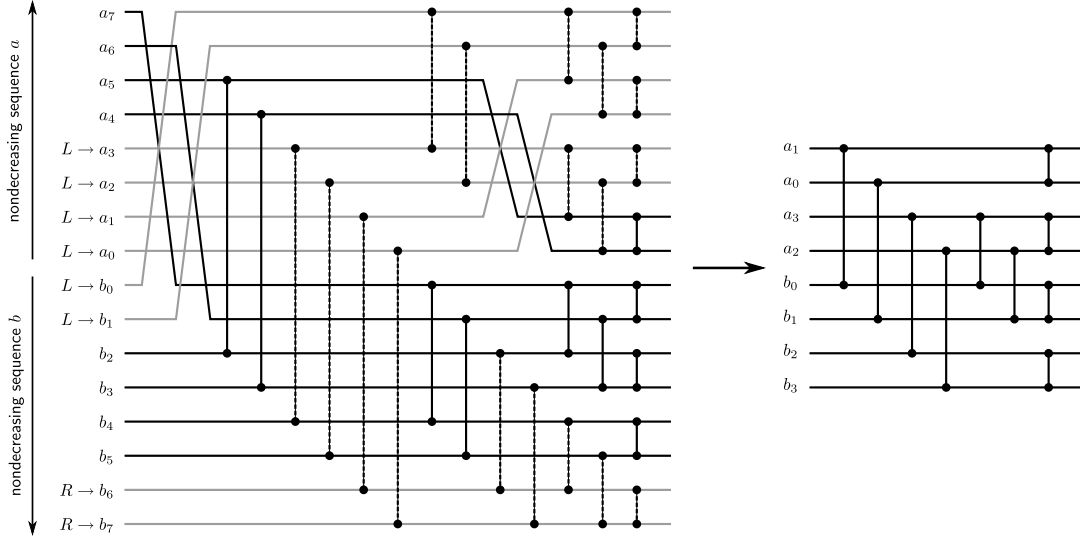


Figure 5: Construction of the $(4,4)$ -improved Bitonic merger M'_8 by wire elimination from the Bitonic $(8,8)$ -merger

and-conquer fashion to build a sorting circuit. However, some care needs to be taken, due to the particular order in which items are expected to be input into the improved bitonic mergers. **Figure 7** shows how the merging circuits must be wired together to create an improved Bitonic sorter IBS_{16} with 16 inputs. The improved Bitonic sorter IBS_n is defined analogously. The twisted wires lead to nonstandard comparators (see **Figure 8**). By Fact 1, we can transform the circuit to a circuit that consists of standard comparators only. It is shown in **Figure 9**.

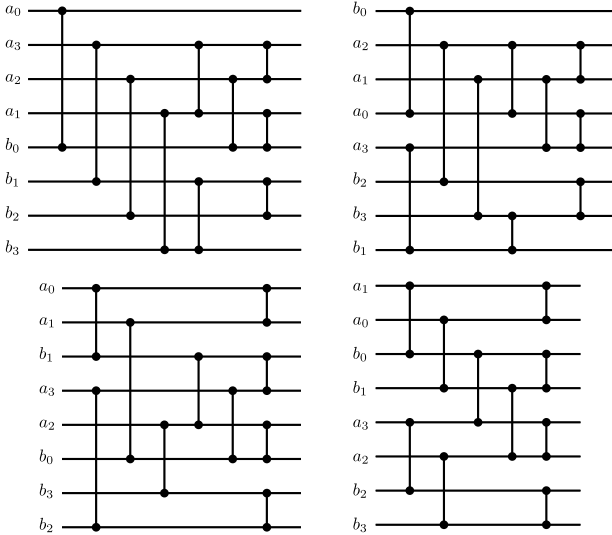


Figure 6: Four more variants of Bitonic $(4,4)$ -Mergers of depth 3 and size 10 obtained by wire elimination from the classical Bitonic $(8,8)$ -merger $BM_{8,8}$, and the corresponding input orders

So we have shown the quintessential result of this paper.

Theorem 4. Let $n = 2^k$. IBS_n is an n -sorter of depth $\frac{1}{2} \log n$ $(\log n + 1)$ and size $\frac{1}{4}n((\log n)^2 + 1)$.

Proof. By the discussion above, IBS_n is an n -sorter, and its depth is equal to the depth of the Bitonic n -sorter BS_n .

Its size $s_{IBS}(n)$ is, with $C(n)$ from Theorem 3,

$$\begin{aligned} s_{IBS}(n) &= 2 \cdot s_{IBS}\left(\frac{1}{2}n\right) + C(n) = \sum_{i=1}^k 2^{k-i} \cdot C(2^i) \\ &= 2^{k-1} + \sum_{i=2}^k 2^{k-i} \cdot \frac{1}{2} \cdot 2^i \cdot \left(i - \frac{1}{2}\right) \\ &= \frac{1}{4} \cdot 2^k \cdot (k^2 + 1) = \frac{1}{4} \cdot n \cdot ((\log n)^2 + 1) . \end{aligned}$$

□

Hence, IBS_n consists of $\frac{1}{4}n(\log n - 1)$ comparators less than the classical Bitonic n -sorter and $\frac{1}{4}n(\log n - 3) + 1$ comparators more than the Odd-Even Merge n -sorter.

5 Concluding Remarks

In our construction of M'_n , we started with the Bitonic (n,n) -merger $BM_{n,n}$ assuming the input order $a_{n-1}, \dots, a_0, b_0, \dots, b_{n-1}$. As $BM_{n,n}$ also correctly merges any cyclic shift of the order mentioned above, even more variants than depicted in Figure 6 can be obtained by wire elimination automatically. However, in the light of the results due to Bilardi [3], many of the merging circuits obtained in this way seem to be structurally equivalent.

References

- [1] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $c \cdot \log n$ parallel steps. *Combinatorica*, 3:1–19, 1983.
- [2] K. E. Batcher. Sorting networks and their applications. In *AFIPS Conf. Proc. 32*, pages 307–314, 1968.
- [3] G. Bilardi. Merging and sorting networks with the topology of the Omega network. *IEEE Trans. Comput.*, 38:1396–1403, 1989.

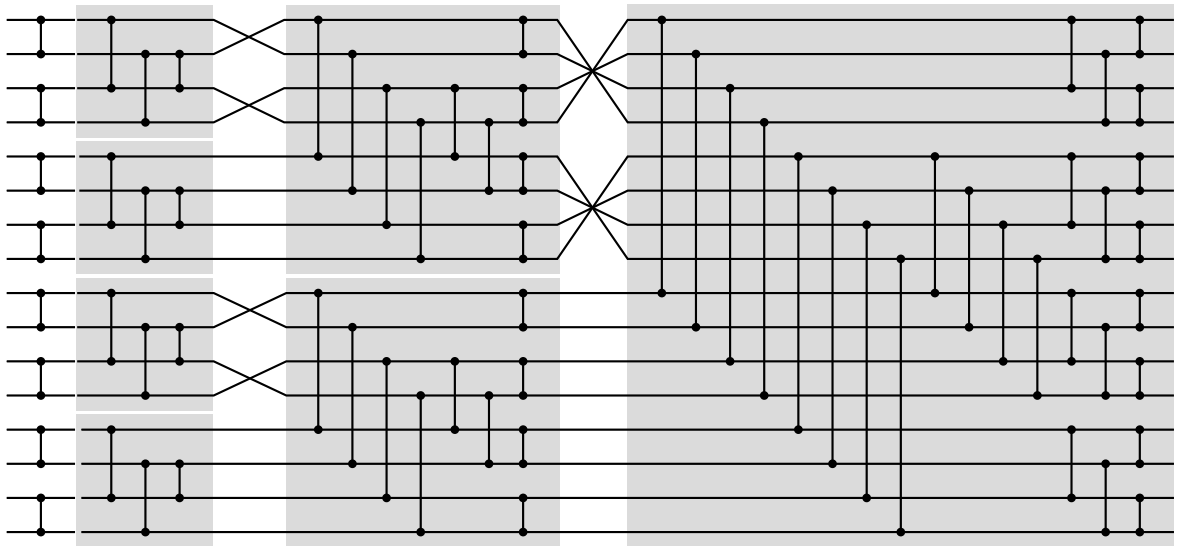


Figure 7: Recursive construction of the improved Bitonic 16-sorter IBS_{16} . The shaded areas denote copies of improved Bitonic mergers. Its depth is 10 and its size is 68, in contrast to BS_{16} whose size is 80.

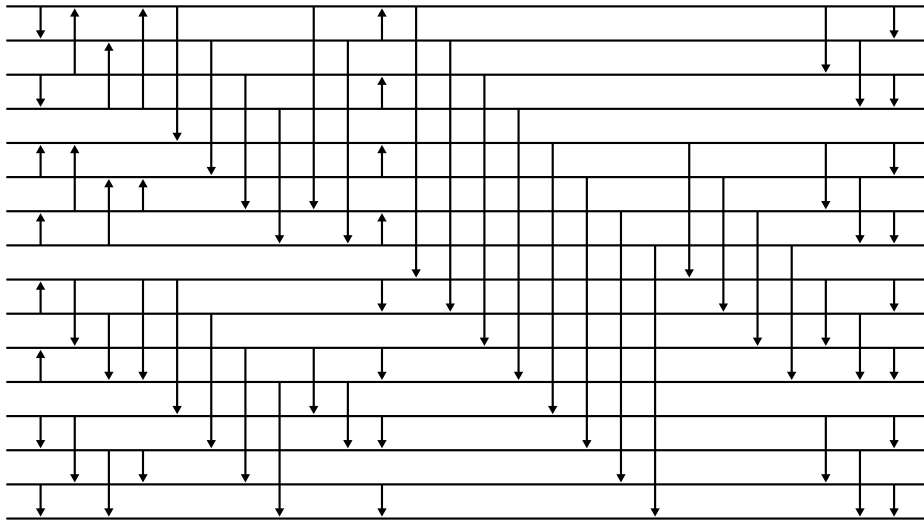


Figure 8: The improved Bitonic 16-sorter IBS_{16} from Figure 7 with straight wires

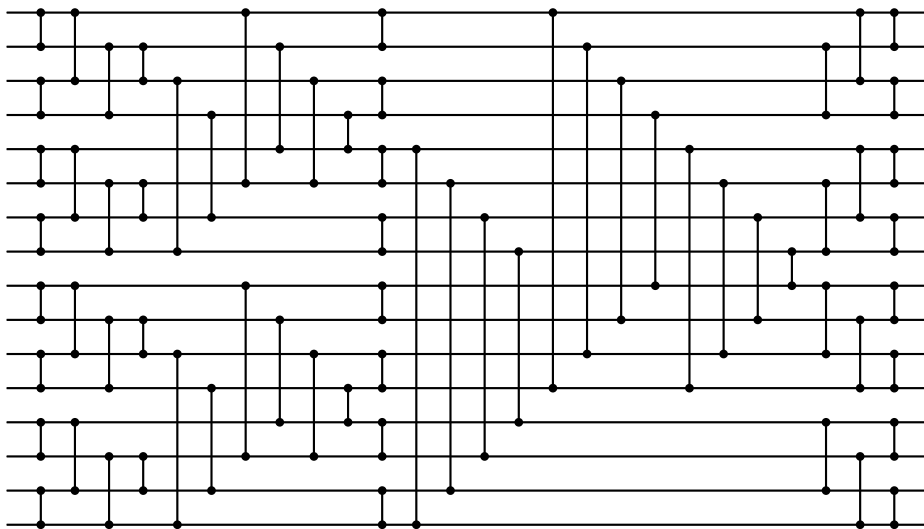


Figure 9: The improved Bitonic 16-sorter IBS_{16} from Figure 8 with standard comparators only

- [4] K. Brockmann and R. Wanka. Efficient oblivious parallel sorting on the MasPar MP-1. In *Proc. 30th Hawaii International Conference on System Sciences (HICSS)*, pages 200–208, Vol. I, 1997.
- [5] Z. Hong and R. Sedgewick. Notes on merging networks. In *Proc. 14th ACM Symp. on Theory of Computing (STOC)*, pages 296–302, 1982.
- [6] D. E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, Reading, Massachusetts, 2nd edition, 1998.
- [7] M. Kutylowski, K. Lorys, B. Oesterdiekhoff, and R. Wanka. Periodification scheme: Constructing sorting networks with constant period. *J. ACM*, 47:944–967, 2000.
- [8] T. Leighton. Tight bounds on the complexity of parallel sorting. *IEEE Trans. Comput.*, 34:344–354, 1985.
- [9] M. S. Paterson. Improved sorting networks with $O(\log n)$ depth. *Algorithmica*, 5:75–92, 1990.
- [10] A. Wachsmann and R. Wanka. Sorting on a massively parallel system using a library of basic primitives: Modeling and experimental results. In *Proc. 3rd European Conference in Parallel Processing (Euro-Par)*, pages 399–408, 1997.