Information Processing Letters 118 (2017) 17–20

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Merging almost sorted sequences yields a 24-sorter

Thorsten Ehlers

University of Kiel, Germany

ARTICLE INFO

Article history:
Received 16 March 2016
Received in revised form 17 August 2016
Accepted 17 August 2016
Available online 20 September 2016
Communicated by M. Chrobak

Keywords:
Parallel processing
Sorting networks
Theory of computation

ABSTRACT

We present a new sorting network on 24 channels, which uses only 12 layers, improving the previously best known bound by one layer. By monotonicity, this also implies improved sorting networks for 23 channels. This result was obtained by combining techniques for generating prefixes of sorting networks with propositional encodings.

1. Introduction

Comparison based sorting algorithms like Quicksort perform a sequence of compare operations which depends on the input data. Contrary to this, data oblivious sorting algorithms perform comparisons in a predefined way. This makes them relevant for parallel sorting algorithms: Any two compare-and-swap operations that are performed on disjoint inputs may be done in parallel, e.g. if implemented on a FPGA, sets of independent comparisons can be done in parallel in one stage of a pipelined design [1]. It is common to present oblivious sorting algorithms as sorting networks [2], cf. Fig. 1.

On the left hand side, the input vector $(5, 4, 3, 2, 1)$ is attached to the horizontal lines, denoted channels. Comparators are shown as vertical lines connecting two channels. Each comparator compares the values on its input channels, and sorts them non-decreasingly. The dashed lines separate layers of the sorting network: all comparators within one layer touch pairwise disjoint channels, therefore they can work in parallel. Thus the number of layers determines the number of parallel sorting steps required to sort an input, so networks with fewer layers are faster.

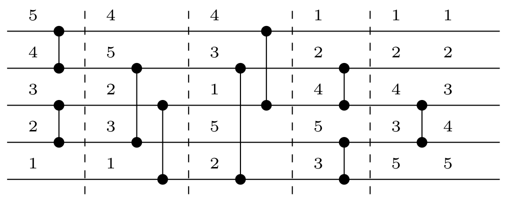

Fig. 1. A sorting network on 5 channels with input $(5, 4, 3, 2, 1)$.

2. Background and related work

Trivial algorithms like Bubble Sort are oblivious, but require $\mathcal{O}(n^2)$ comparisons. Batcher suggested two ways of construction sorting networks, Odd–Even–Mergesort and Bitonic Mergesort [2,3], both with $\mathcal{O}(n \log^2(n))$ comparisons in $\mathcal{O}(\log^2(n))$ layers. This asymptotic bound was improved by Ajtai, Komlós and Szemerédi, who showed that oblivious sorting algorithms exist which only need $\mathcal{O}(n \log(n))$ comparisons in $\mathcal{O}(\log(n))$ layers [4], which is asymptotically optimal. Unfortunately, the constants hidden in the $\mathcal{O}$-notation are huge, which makes Batchers sorting networks superior for every practical number of inputs.

There exist different metrics for describing properties of sorting networks. The size of a sorting network measures the number of comparators required, and is lower-

E-mail address: the@informatik.uni-kiel.de.

http://dx.doi.org/10.1016/j.ipl.2016.08.005
0020-0190/© 2016 Elsevier B.V. All rights reserved.

Table 1
Optimal depth (d_n) of sorting networks on n inputs, for $n \leq 12$.

n	1	2	3	4	5	6	7	8	9	10	11	12
d_n	0	1	3	3	5	5	6	6	7	7	8	8

Table 2
Best known values and bounds on optimal depth (d_n) of sorting networks on n inputs, for $13 \leq n \leq 24$. The contributions of this paper are shown in boldface. Note that the new result for $n = 24$ also implies one for $n = 23$: Removing one channel, and all connected comparators, yields a sorting network on 23 channels.

n	13	14	15	16	17	18	19	20	21	22	23	24
d_n	9	9	9	9	10	11 10	11 10	11 10	12 10	12 10	**12** 10	**12** 10

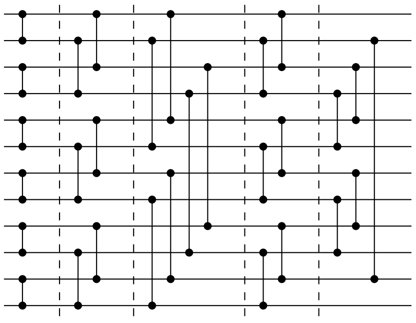
Fig. 2. Prefix of a sorting network on 12 channels, and 5 layers.

bounded by $\mathcal{O}(n \log(n))$, as sorting networks sort based on comparisons. Optimal sorting networks are known only for at most 10 inputs [5], some upper bounds can be found in [6].

This paper focusses on the depth of sorting networks rather than their size. In 1973, Knuth summarised upper bounds on the depth of sorting networks on $n \leq 16$ channels [7], cf. Table 1 and 2. In 1989, Parberry used a SAT-based approach together with a symmetry break on the first layer to prove that the bounds for $n \leq 10$ are optimal [8]. This was pushed further by Bundala and Závodný in 2014 [9]. Using a decomposition and symmetry breaking approach for the first two layers combined with SAT solving, they were able to prove that the bounds for $n \leq 16$ are optimal. Al-Baddar and Batcher developed a tool to analyse prefixes, i.e. some layers on the left hand side of a sorting network, which allowed them to hand-craft improved sorting networks for 18 and 22 channels [10]. Ehlers and Müller used a SAT solver to extend handcrafted prefixes, and found faster sorting networks for 17, 19 and 20 channels [11].

The SAT encodings used were not strong enough to prove optimality for any new case. Codish et al. introduced symmetry breaks for the last layers of a sorting network [12]. Ehlers and Müller suggested an improved SAT encoding, and re-ordered the channels of sorting networks, which can be used to reduce the number of variables in the SAT encoding, allowing to prove that 10 layers are optimal when sorting 17 inputs [13].

The purpose of this paper is to show this combination of techniques, and the presentation of better sorting networks on 24 channels. For details on the propositional encoding of sorting networks we refer to [9,13]. Techniques to generate sets of prefixes up to symmetries can be found in [9,5].

3. Construction of new sorting networks

The sorting networks suggested by Batcher can be constructed algorithmically [3], but they are not optimal for $n > 8$ channels. The sorting networks for $n > 8$ shown in [2] are handcrafted, and can be used as base cases for merging based algorithms.

Bundala and Závodný generated sets of Pareto-optimal prefixes on 2 layers and checked, using a SAT solver, which of these can be extended to a sorting network of some depth. Here, one prefix p_1 is considered superior to another prefix p_2 if every sorting network beginning with p_2 can be transformed into one beginning with p_1. In [11], Ehlers and Müller handcrafted prefixes, mainly based on so-called green filters [14] and used a SAT solver to extend these to a full sorting network.

All these approaches are somewhat limited: Handcrafting sorting networks is limited by the ability of a human to understand sorting networks. Current SAT-based approaches do not scale well, and generating all prefixes yields huge sets to test, even when symmetries are considered [5]. We therefore used a combination of these approaches. First, we generate the prefix of a sorting network on 12 channels which almost sorts its inputs. As it seems intractable to consider all such prefixes, we use a greedy approach: given some prefixes on k layers, we generate all Pareto-optimal prefixes on $k+1$ layers up to symmetries, and keep only the 32 offsprings which yield a minimum number of outputs. Iterating this process gives the prefix on 5 layers shown in Fig. 2, which has 34 different output vectors.

Next, we create a prefix on 24 channels consisting of two prefixes on 12 channels, and add two comparators to their last layer which connect unused channels. This gives a total of 1,129 outputs which remain to be sorted by the remainder of the network, which is a tractable size for a SAT solver. This prefix was permuted to minimise the SAT encoding used lateron. The formula to solve has 56,949 variables and 1,164,158 clauses, and can be solved by MiniSAT [15] in less than 7 hours on an Intel i7-4770HQ CPU.

The generated sorting network is presented in Fig. 3. Due to the permutation of the channels, it is hard to understand its structure. Therefore, we present an alternative version in Fig. 4. Here, we permuted the channels such that the original prefix was restored, and removed redundant comparators. In this presentation, the structure in the first 5 layers becomes visible again. Interestingly, the 6th layer, which was generated by the SAT solver, is very similar to the first layers of a merge step in Batcher's construction [2].

The sorting network in Fig. 4 has 125 comparators. Although improving the upper bound on the depth, it does not improve the upper bounds on the size of sorting networks, as networks with 123 and 118 comparators for 24 and 23 inputs, respectively, are known [6].