A Benchmark for the Sorting Network Problem

Michal Bidlo
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic
bidlom@fit.vutbr.cz

Categories and Subject Descriptors

B.2.2 [**Arithmetic and Logic Structures**]: Performance Analysis and Design Aids; B.8.0 [**Performance and Reliability**]: General; B.6.1 [**Logic Design**]: Design Styles

General Terms

Design, Performance

Keywords

Sorting network, benchmark

1. SORTING NETWORKS CONCEPT

Consider a *compare–swap* operation that compares and possibly swaps the values of its two operands (a, b), so that we obtain a pair (a, b) satisfying $a \leq b$ after execution. A *sorting network* is a sequence of compare–swap operations that depends only on the number of elements to be sorted [1]. Let's call the compare–swap operation as a comparator. An advantage of sorting networks against the classical sorting algorithms is that the number of comparators is fixed for a given number of inputs. Thus they can be easily implemented in hardware. Figure 1 shows an example of a three-input sorting network and its alternative symbol.

The two main aspects determining the quality of sorting network are the number of comparators (the fewer comparators, the lower implementation cost) and delay (the lower delay, the faster sorting). Let's define the delay of a sorting network as the number of groups of independent comparators (i.e. the groups of comparators, whose input indices are mutually different). Such the comparators may be performed in parallel. We denote a comparator of a sorting network as *redundant* if it does not swap any input values during the complete test of the sorting network. Such a comparator can be removed from the sorting network without any loss of functionality.

2. BENCHMARK SORTING NETWORKS

GECCO'05, June 25–29, 2005, Washington, DC, USA.

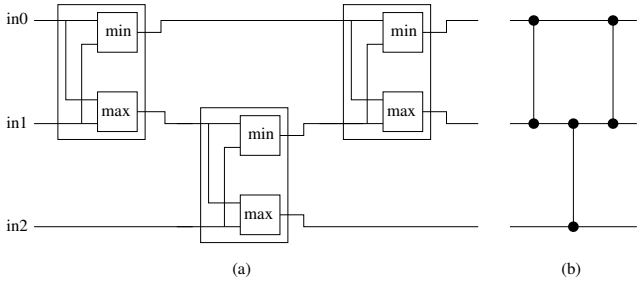

Figure 1: (a) A 3-input sorting network consists of 3 comparators, each of which contains components for computing minumim and maximum. (b) Alternative symbol.

We gained a program using evolutionary algorithm for the construction of arbitrarily large sorting networks from a simple instance of the problem (an embryo). The evolved program creates sorting networks with only even number of inputs. The number of comparators of the sorting networks built by means of this program is substantially lower than the number of comparators of conventionally-constructed sorting networks (e.g. bubble-sort networks). Although the evolved program creates sorting networks with some redundant comparators, after removing them we gain sorting networks with substantially better properties (both the number of comparators and delay) in comparison with a conventional solution.

Table 1 contains the number of comparators and Table 2 contains delay values of selected even-input sorting networks. Table 3 summarizes the properties of resulting sorting networks (after removing the redundant comparators). We determined the number of comparators and delay of the conventional sorting networks by analyzing bubble-sort networks published in [1]. Figure 2 shows the structure of the proposed sorting networks.

Note, that after removing the bottom input line of the sorting network with all the comparators connected to that input line we obtain a valid sorting network with odd number of inputs. Although the sorting networks constructed using the evolved program were tested only up to 28 inputs, we believe the program is able to construct arbitrarily large networks (formal proof is the objective of our future work). In addition, the proposed circuits can be also considered as median networks.

N	6	8	10	12	14	16	18	20	22	24	26	28
conventional	15	28	45	66	91	120	153	190	231	276	325	378
evolved	13 (1)	24 (2)	38 (3)	55 (4)	75 (5)	98 (6)	124 (7)	153 (8)	185 (9)	220 (10)	258 (11)	299 (12)

Table 1: The number of comparators of even-input sorting networks created by means of the evolved program in comparison with conventional sorting networks. Numbers in parentheses denote the number of redundant comparators of the sorting networks.

N	6	8	10	12	14	16	18	20	22	24	26	28
konv.	9	13	17	21	25	29	33	37	41	45	49	53
evolved	6 (6)	9 (9)	14 (12)	19 (15)	23 (18)	26 (21)	31 (24)	36 (27)	41 (30)	46 (33)	51 (36)	56 (39)

Table 2: Delay of even-input sorting networks created by means of the evolved program in comparison with conventional sorting networks. Numbers in parentheses denote the delay of the resulting sorting networks (after removing the redundant comparators).

N	6	8	10	12	14	16	18	20	22	24	26	28
conventional	15 (9)	28 (13)	45 (17)	66 (21)	91 (25)	120 (29)	153 (33)	190 (37)	231 (41)	276 (45)	325 (49)	378 (53)
evolved	12 (6)	22 (9)	35 (12)	51 (15)	70 (18)	92 (21)	117 (24)	145 (27)	176 (30)	210 (33)	247 (36)	287 (39)

Table 3: Summary of the properties of resulting even-input sorting networks in comparison with conventional sorting networks. Numbers without parentheses denote the humber of comparators and numbers in parentheses denote the delay of resulting sorting networks. These sorting networks do not contain redundant comparators.

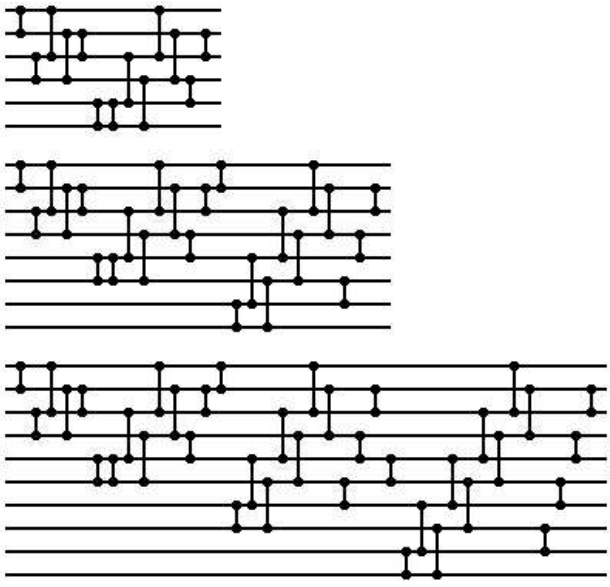

Figure 2: Structure of the sorting networks with 6, 8 and 10 inputs created by means of the evolved program. These sorting networks contain some redundant comparators.

3. ACKNOWLEDGMENTS

The research was performed with the support of the Grant Agency of the Czech Republic under No. 102/04/0737 *Modern Methods of Digital Systems Design*. M.Bidlo was partially supported from J. Hlavka foundation.

4. REFERENCES

[1] D. E. Knuth. *The Art of Computer Programming: Sorting and Searching (2nd ed.)*. Addison Wesley, 1998.