CSE-40533

Introduction to Parallel Processing

Chapter 7

Sorting and Selection Networks

- **Become familiar with the circuit-level models of parallel processing**
- **Introduce useful design tools and trade-off issues via a familiar problem**

7.1 What Is a Sorting Network?

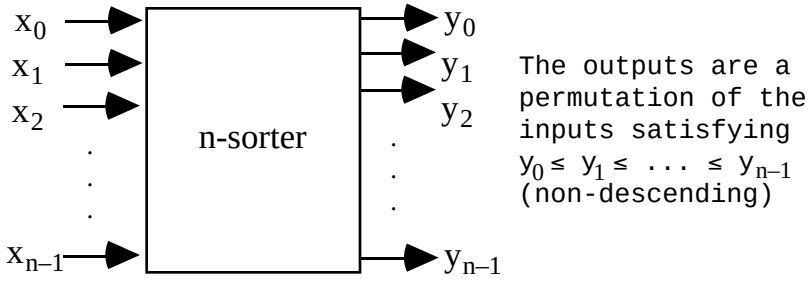


Fig. 7.1. An n-input sorting network or an n-sorter.

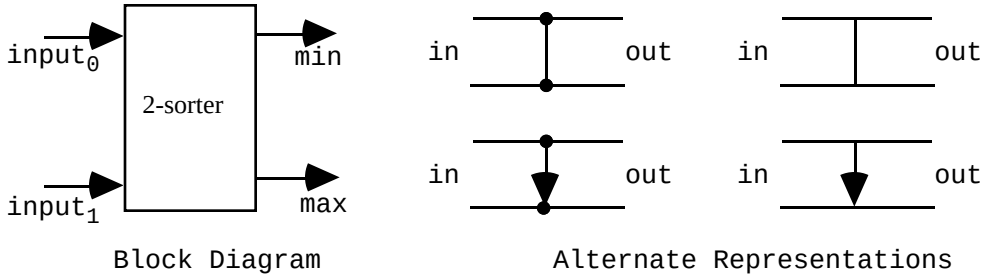


Fig. 7.2. Block diagram and four different schematic representations for a 2-sorter.

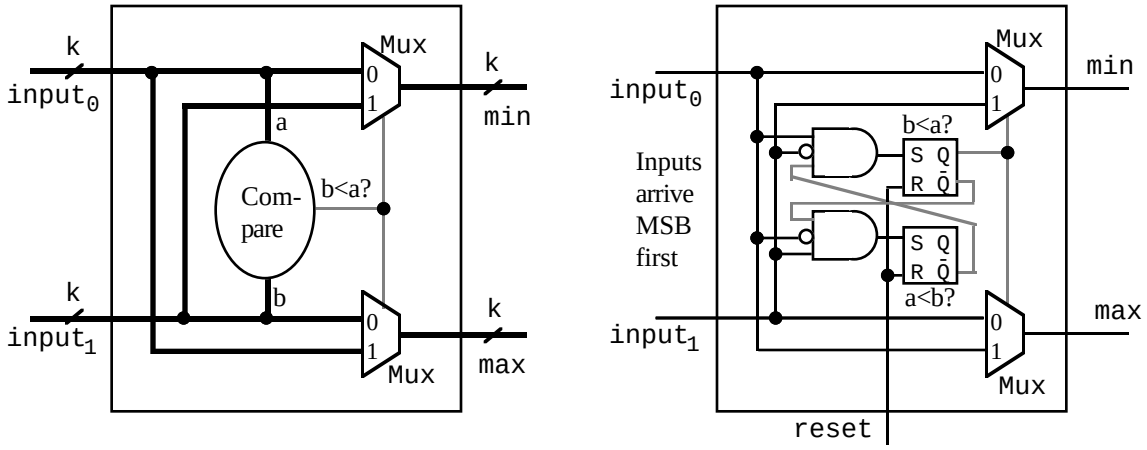


Fig. 7.3. Parallel and bit-serial hardware realizations of a 2-sorter.

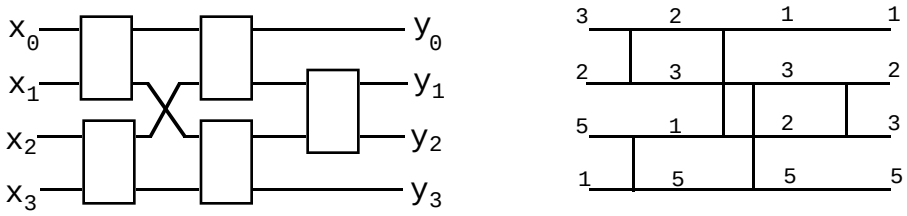


Fig. 7.4. Block diagram and schematic representation of a 4-sorter.

How to verify that the circuit of Fig. 7.4 is a valid 4-sorter?

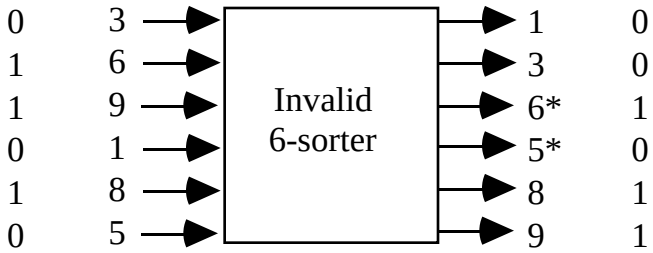
The answer is easy in this case

After the first two circuit levels, the top line carries the smallest and the bottom line the largest of the four values

The final 2-sorter orders the middle two values

More generally, we need to verify the correctness of an n-sorter through formal proofs or by time-consuming exhaustive testing. Neither approach is attractive.

The zero-one principle: A comparison-based sorter is valid iff it correctly sorts all 0/1 sequences.



7.2 Figures of Merit for Sorting Networks

a. Cost: number of 2-sorter blocks used in the design

b. Delay: number of 2-sorters on the critical path

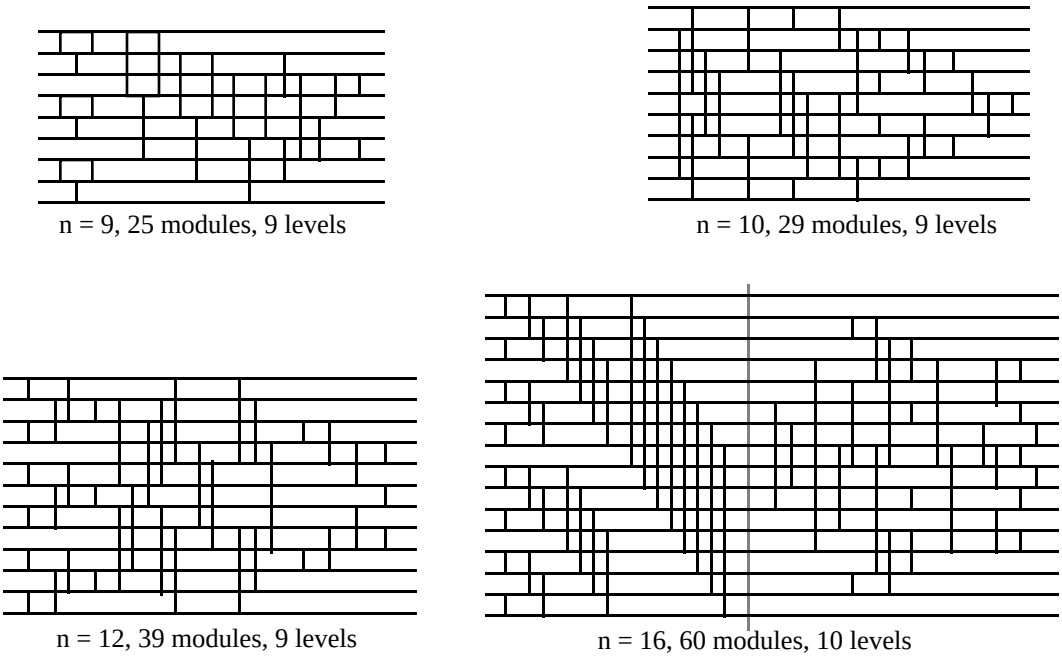


Fig. 7.5. Some low-cost sorting networks.

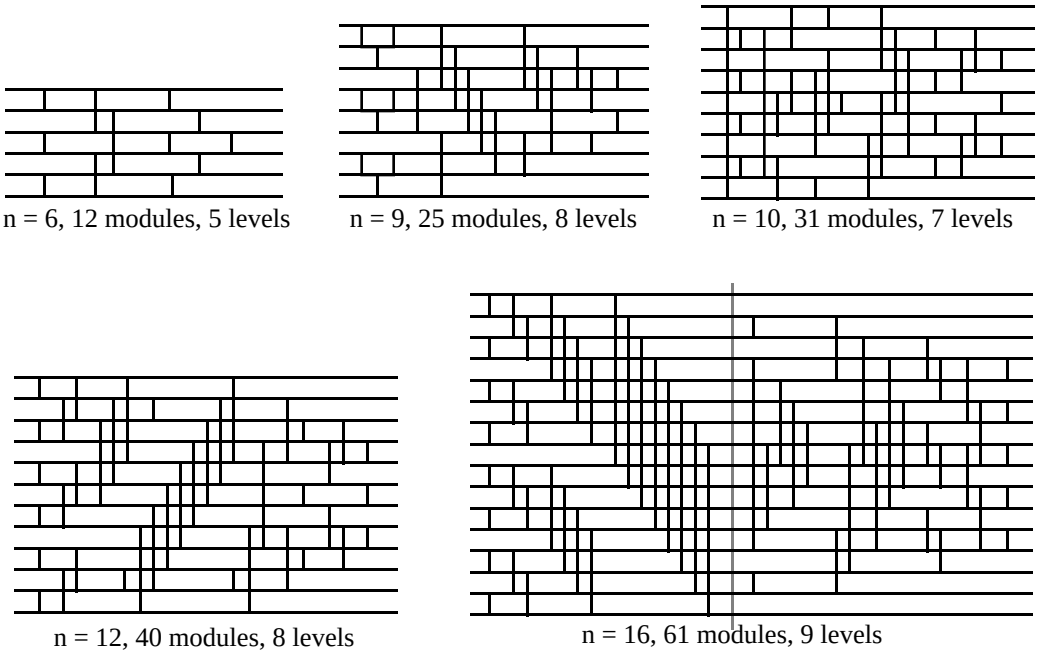


Fig. 7.6. Some fast sorting networks.

7.3 Design of Sorting Networks

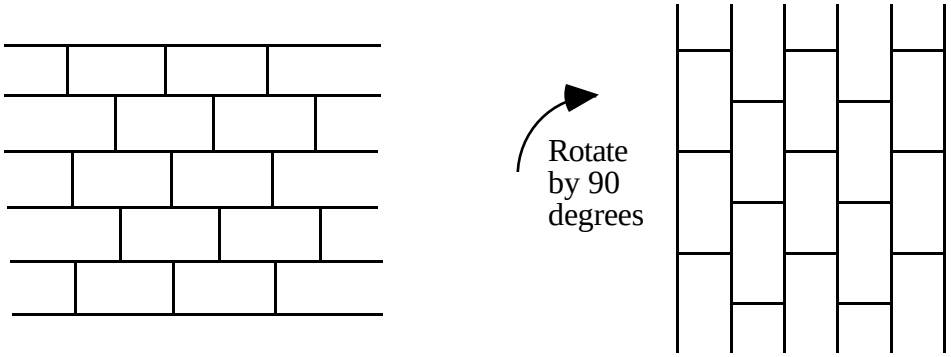
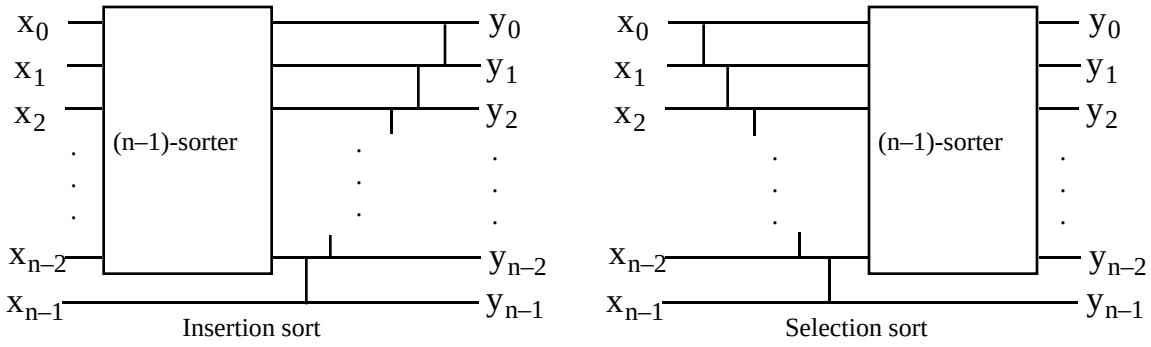


Fig. 7.7. Brick-wall 6-sorter based on odd–even transposition.

$C(n) = C(n-1) + n - 1 = (n-1) + (n-2) + \cdots + 2 + 1 = n(n-1)/2$

$D(n) = D(n-1) + 2 = 2 + 2 + \cdots + 2 + 1 = 2(n-2) + 1 = 2n - 3$

$\text{Cost} \times \text{Delay} = n(n-1)(2n-3)/2 = \Theta(n^3)$



Parallel insertion sort = Parallel selection sort = Parallel bubble sort!

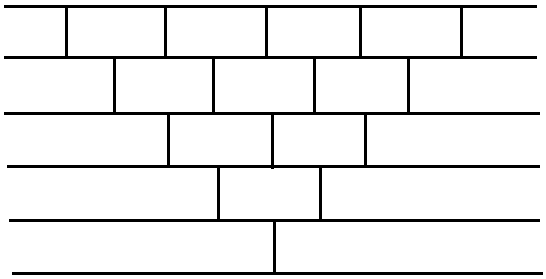
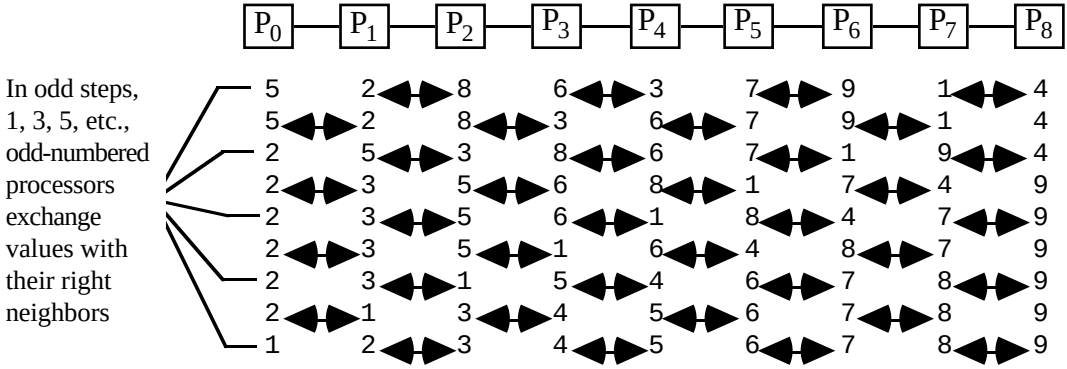


Fig. 7.8. Sorting network based on insertion sort or selection sort.


P0
P1
P2
P3
P4
P5
P6
P7
P8
In odd steps, 1, 3, 5, etc., odd-numbered processors exchange values with their right neighbors
5 2 8 6 3 7 9 1 4
5 2 8 3 6 7 9 1 4
2 5 3 8 6 7 1 9 4
2 3 5 6 8 1 7 4 9
2 3 5 6 1 8 4 7 9
2 3 5 1 6 4 8 7 9
2 3 1 5 4 6 7 8 9
2 1 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9

7.4 Batcher Sorting Networks

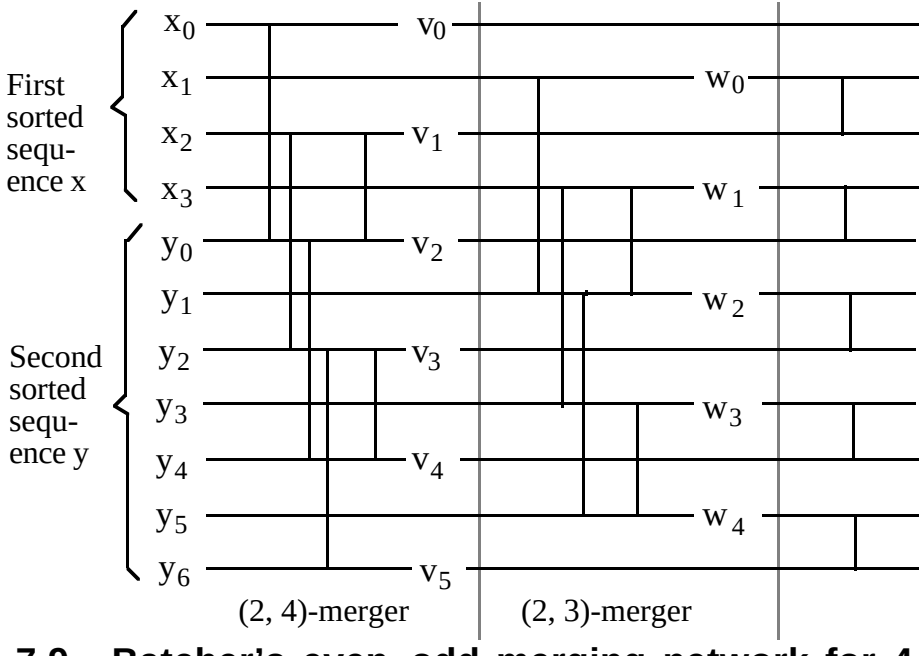


Fig. 7.9. Batcher's even–odd merging network for 4 + 7 inputs.

$x_0 \le x_1 \le \cdots \le x_{m-1}$ (k 0s) $y_0 \le y_1 \le \cdots \le y_{m'-1}$ (k' 0s)

Merge $x_0, x_2, \cdots$ and $y_0, y_2, \cdots$ to get $v_0, v_1, \cdots$ $k_{even} = \lceil k/2 \rceil + \lceil k'/2 \rceil$ 0s

Merge $x_1, x_3, \cdots$ and $y_1, y_3, \cdots$ to get $w_0, w_1, \cdots$ $k_{odd} = \lfloor k/2 \rfloor + \lfloor k'/2 \rfloor$ 0s

Compare-exchange the pairs of elements $w_0{:}v_1, w_1{:}v_2, w_2{:}v_3, \cdots$

Batcher's (m, m) even-odd merger, when m is a power of 2, is characterized by the following recurrences:

$C(m) = 2C(m/2) + m - 1 = (m-1) + 2(m/2-1) + 4(m/4-1) + \cdots = m \log_2 m + 1$

$D(m) = D(m/2) + 1 = \log_2 m + 1$

$\text{Cost} \times \text{Delay} = \Theta(m \log^2 m)$

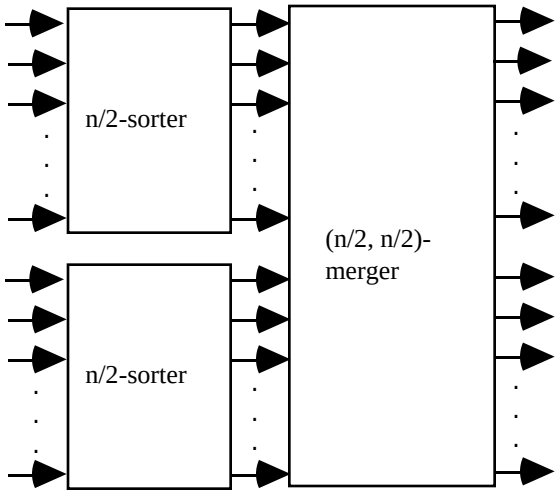


Fig. 7.10. The recursive structure of Batcher's even–odd merge sorting network.

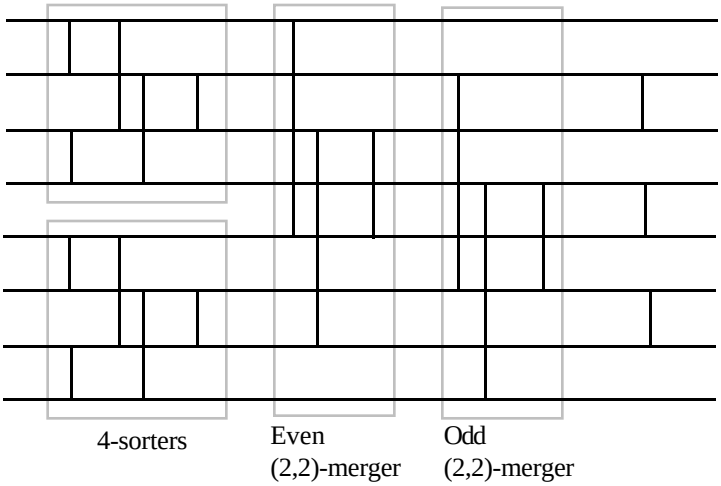


Fig. 7.11. Batcher's even-odd merge sorting network for eight inputs.

Batcher sorting networks based on the even-odd merge technique are characterized by the following recurrences:

$C(n) = 2C(n/2) + (n/2)(\log_2(n/2)) + 1 \approx n(\log_2 n)^2/2$

$D(n) = D(n/2) + \log_2(n/2) + 1 = D(n/2) + \log_2 n = \log_2 n (\log_2 n + 1)/2$

Cost × Delay = $\Theta(n \log^4 n)$

Bitonic sorters

Bitonic sequence: "rises then falls", "falls then rises", or is obtained from the first two categories through cyclic shifts or rotations. Examples include:

1 3 3 4 6 6 6 2 2 1 0 0 Rises, then falls

8 7 7 6 6 6 5 4 6 8 8 9 Falls, then rises

8 9 8 7 7 6 6 6 5 4 6 8 The previous sequence, right-rotated by 2

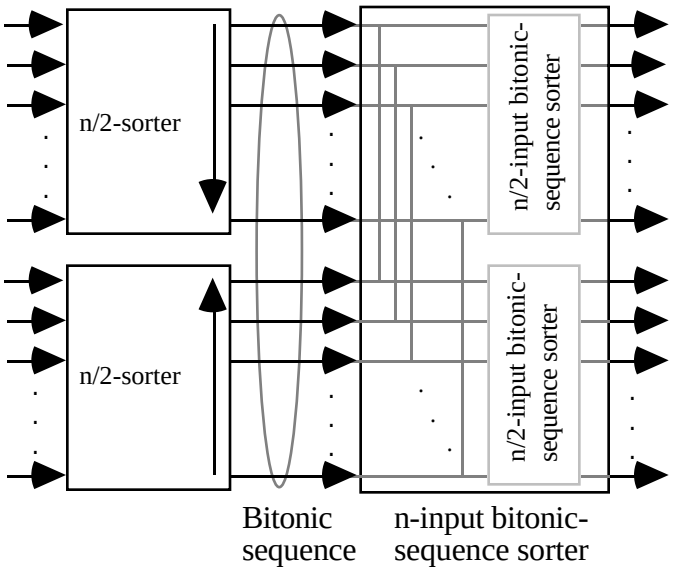
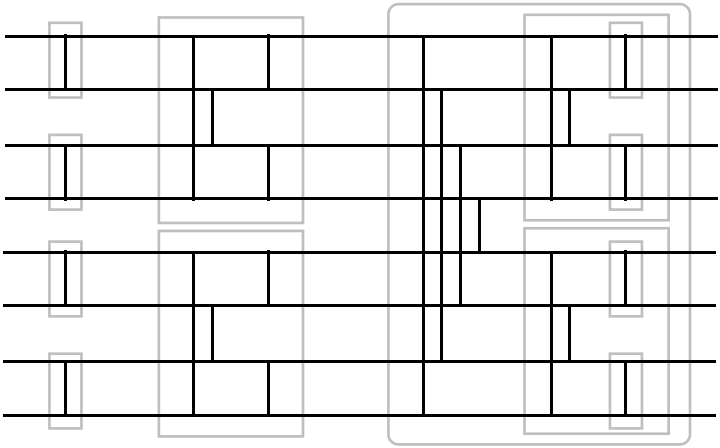


Fig. 7.12. The recursive structure of Batcher's bitonic sorting network.


2-input sorters
4-input bitonic-sequence sorters
8-input bitonic-sequence sorter