Parallel ("Batcher") Sorting

1. Shuffle-exchange interconnection topology
   - $n = 2^k$ processors
   - Each processor has a $k$ bit address $a_1 \cdots a_k$
   - The exchange connection: connect $a_1 \cdots a_k$ to $a_1 \cdots \bar{a}_k$.
   - The shuffle connection: connect $a_1 \cdots a_k$ to $a_2 \cdots a_k a_1$.

   - Picture:

2. Batcher sort (1968)
   - Definition: a sequence $x_0, \ldots, x_{n-1}$ is \textit{bitonic} if $\exists$ index $j$ s.t. $x_0, \ldots, x_j$ is increasing and $x_{j+1}, \ldots, x_{n-1}$ is decreasing, or a cyclic shift generates such a sequence.
• Key observation: if \( x_0, \ldots, x_{n-1} \) is bitonic and if
\[
\begin{align*}
y_i & \equiv \min(x_i, x_{i+n/2}) \\
z_i & \equiv \max(x_i, x_{i+n/2})
\end{align*}
\]
for \( i = 0, \ldots, n/2 - 1 \), then the sequences \( y \) and \( z \) are bitonic, and \( y_i < z_j \) for all \( i, j \).

• We can repeat this recursively.
• A bitonic sorter.

• Why does this work?
  \begin{itemize}
  \item At first stage, compare numbers in PE \( 0a_2 \cdots a_k \) and \( 1a_2 \cdots a_k \). But
  \[
  \begin{align*}
  Sh(0a_2 \cdots a_k) &= a_2 \cdots a_k 0 \\
  Sh(1a_2 \cdots a_k) &= a_2 \cdots a_k 1
  \end{align*}
  \]
  \item So after the first exchange, the small numbers (\( y \)'s) are in PEs \( a_2 \cdots a_k 0 \), and large numbers (\( z \)'s) are in PEs \( a_2 \cdots a_k 1 \).
  \item Now we need to compare numbers in PE \( 0a_3 \cdots a_k 0 \) and \( 1a_3 \cdots a_k 0 \). But
  \[
  \begin{align*}
  Sh(0a_3 \cdots a_k 0) &= a_3 \cdots a_k 00 \\
  Sh(1a_3 \cdots a_k 0) &= a_2 \cdots a_k 01
  \end{align*}
  \]
  and similarly
  \[
  \begin{align*}
  Sh(0a_3 \cdots a_k 1) &= a_3 \cdots a_k 10 \\
  Sh(1a_3 \cdots a_k 1) &= a_2 \cdots a_k 11
  \end{align*}
  \]
  \end{itemize}
– After \( k - 1 \) stages:
  PEs \( a_k0\cdots0 \) have smallest 2 \( x \)'s
  PEs \( a_k0\cdots1 \) have next smallest 2 \( x \)'s
  :
  PEs \( a_k1\cdots1 \) have largest 2 \( x \)'s
– So one more shuffle and one more exchange will sort the list.

• Combining bitonic sorters to do sorting of completely unsorted data.
  – Idea:
    * unsorted list is \( n/2 \) bitonic sequences of length 2.
    * merge to get \( n/4 \) bitonic sequences of length 4, etc.
    * every other sequence must be decreasing.
  – Picture

– Time complexity: \( O(\log^2(n)) \)