Combinational Circuits and Sorting Networks

Combinational Circuits:

1. were among earliest developed in terms of a systematic study of parallel algorithms
2. they are simple - so it is a good place to start a semester
3. many such algorithms serve as the basis for models that we will discuss later in the semester

Characteristics of Combinational Circuits:

1. Takes a number of inputs from one end and produces a number of inputs at the other end (typically viewed as left to right)
2. Processors are simple - each performs a single operation in *unit time*. Examples include:
    - a) logical
        - i) and
        - ii) or
        - iii) not
        - iv) Comparison
            - a) <
            - b) >
    - b) arithmetic
        - i) addition
        - ii) subtraction
3. Processors are connected by *unidirectional* links
4. Processors have constant *fan-in* and constant *fan-out*
5. No processor is used more than once (i.e., no feedback)

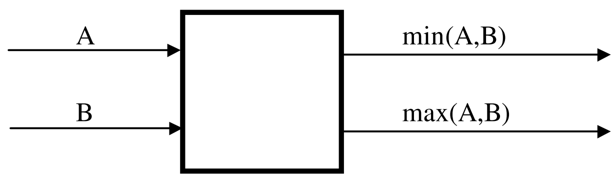
Sorting Networks

Sorting networks are *oblivious* to their inputs. That is, the sequence of comparisons performed is fixed in advance - not a function of the input values. Notice that this is not true for some traditional sorting routines, such as Quicksort or Heapsort, which perform comparisons dependent on the input.

Bitonic Sort: K.E. Batcher, Proc. Spring Joint Computer Conference, 1968, pp. 307-314.

Originally defined in terms of sorting networks to not only sort, but to be used as a multiple input - multiple output switching network for routing.

The basic element of a sorting network is the comparison element, which receives two inputs, A and B, and produces both their minimum and maximum as output.



Definition: A sequence $z_1, z_2, \ldots, z_p$ of p numbers is bitonic if

1. $z_1 \le z_2 \le \ldots \le z_k \ge \ldots \ge z_p$ or
2. $z_1 \ge z_2 \ge \ldots \ge z_k \le \ldots \le z_p$ or
3. if z is split into two parts and the two parts are interchanged to give either (1) or (2) {note: with this, (1) and (2) are equivalent}

Observation: Two *monotone* sequences can be concatenated to form a bitonic sequence. Therefore, a network that sorts a bitonic sequence into monotonic order can be used as a *merging* network.

Theorem: Given a **bitonic** sequence $\langle a_1, a_2, \ldots, a_{2n} \rangle$

a) $d = \langle \min(a_1, a_{n+1}), \min(a_2, a_{n+2}), \ldots, \min(a_n, a_{2n}) \rangle$ is bitonic

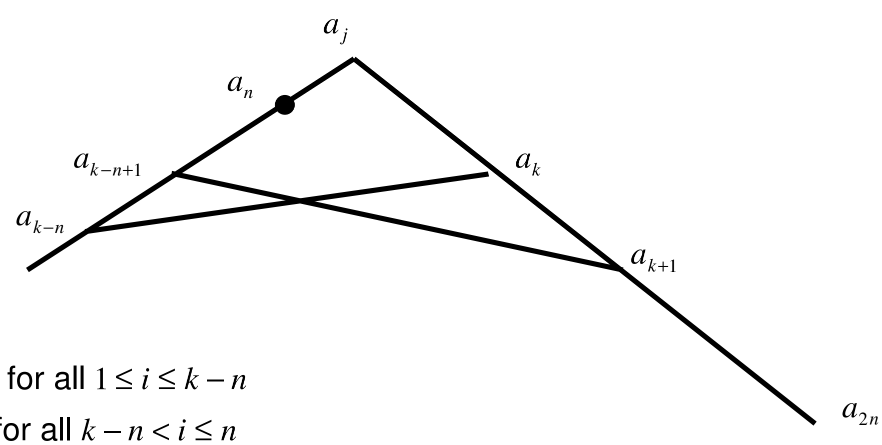
b) $e = \langle \max(a_1, a_{n+1}), \max(a_2, a_{n+2}), \ldots, \max(a_n, a_{2n}) \rangle$ is bitonic

c) $\max(d) \le \min(e)$

Proof: Let $\langle a_a, \ldots, a_{2n} \rangle$ be bitonic. Let $d_i = \min(a_i, a_{n+i}), e_i = \max(a_i, a_{n+i}), 1 \le i \le n$.
We must prove that i) d and e are bitonic <u>and</u> ii) max(d)≤ min(e).
WLOG, $a_1 \le a_2 \le \ldots \le a_{j-1} \le a_j \ge a_{j+1} \ge \ldots \ge a_{2n}, n < j \le 2n$

A) If $a_n \le a_{2n} \Rightarrow a_i \le a_{n+i}$, so $d_i = a_i, e_i = a_{n+i}, 1 \le i \le n$, so DONE.

B) If $a_n > a_{2n}$ then since $a_{j-n} \le a_j$ we know that there exists a k such that $j \le k < 2n$, $a_{k-n} \le a_k$ <u>and</u> $a_{k-n+1} > a_{k+1}$. Note: $\langle a_j, a_{j+1}, \ldots, a_{2n} \rangle$ is decreasing while $\langle a_{j-n}, a_{j-n+1}, \ldots, a_n \rangle$ is increasing.



then $d_i = a_i, e_i = a_{i+n}$, for all $1 \le i \le k-n$
<u>and</u> $d_i = a_{i+n}, e_i = a_i$ for all $k-n < i \le n$

So:

$d_i \le d_{i+1}$ for $1 \le i < k-n$

$d_i \ge d_{i+1}$ for $k-n < i < n$

$e_i \le e_{i+1}$ for $k-n < i < n$

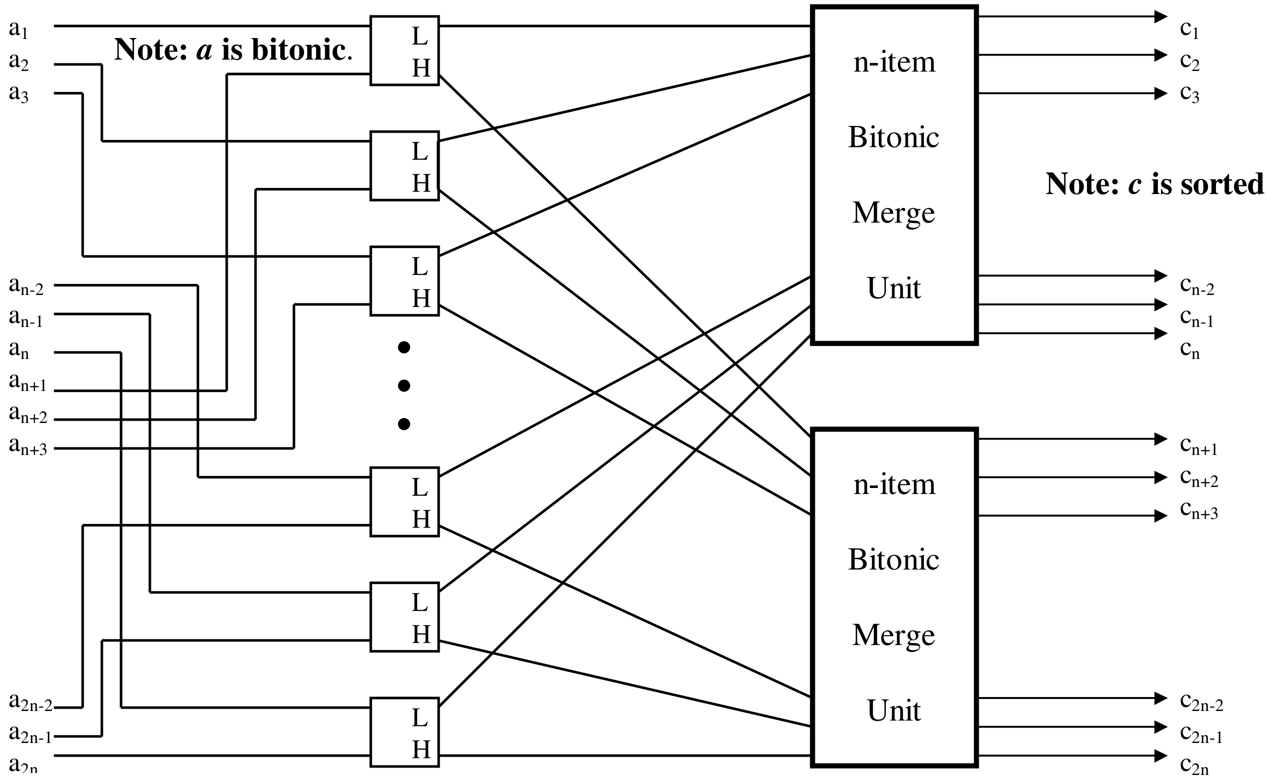
$e_n \le e_1$

$e_i \le e_{i+1}$ for $1 \le i < j-n$

$e_i \ge e_{i+1}$ for $j-n \le i < k-n$

Therefore, d and e are bitonic <u>and</u> $\max(d) = \max(a_{k-n}, a_{k+1}) \le \min(a_k, a_{k-n+1}) = \min(e)$. ■

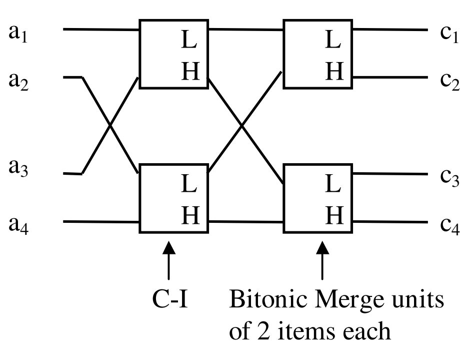
This gives the iterative rule for constructing a bitonic **merge** unit:



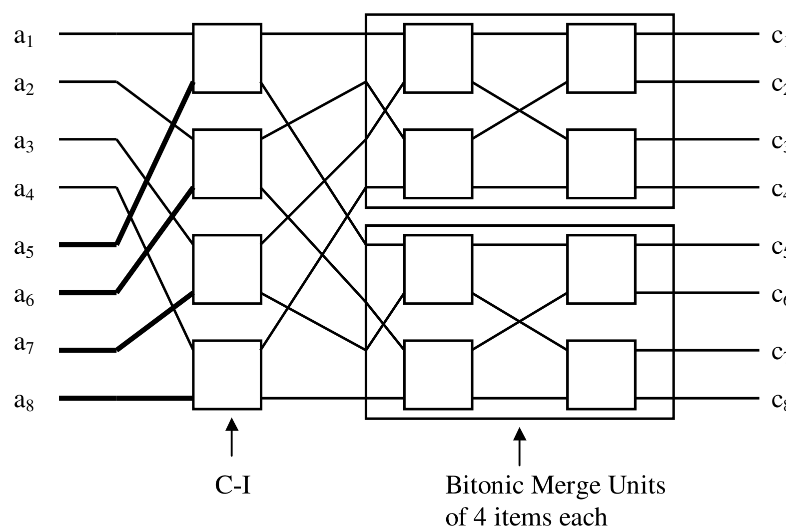
A bitonic merge unit for $2n$ numbers can be constructed from n comparators and two n-item bitonic merge units.

So, to sort a bitonic sequence of size $2n$ requires $L(2n) = L(n) + 1 = \log_2 2n$ levels ($L(2) = 1$)

Examples: note that a is bitonic



Levels = L(2n) =L(2*2) = 1+L(n)
= 1+L(2)
= 2
= $\log_2$(2*2) = 2

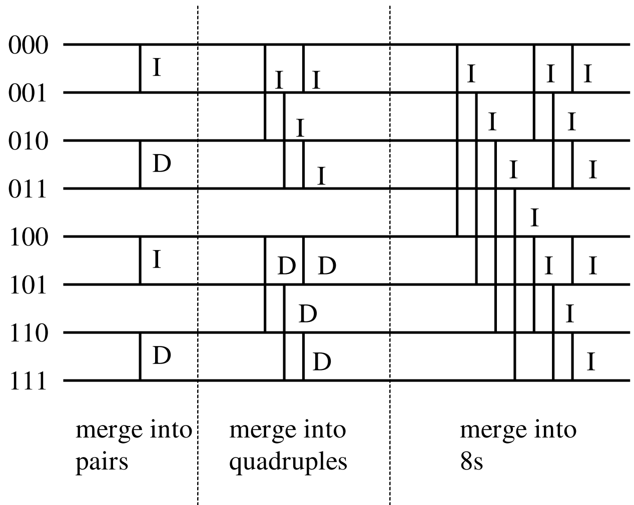


Levels = L(2n) = L(2*4) = 1 + L(4) = 1 + 2 = 3
= $\log_2$(2*4) = 3

Sorting: Build bottom-up using bitonic merge units for merging a bitonic sequence into a sorted list. To sort $2n$ elements, initially presented in an arbitrary order, requires $\Theta(\log^2 n)$ levels and $\Theta(n \log^2 n)$ comparators.

So, sorting $= \sum_{i=1}^{\log_2 2n} i = \frac{(\log_2 2n)(\log_2 2n + 1)}{2}$ levels $= \frac{\log^2(2n)}{2} + \frac{\log(2n)}{2}$

Another view: Sorting 8 random values into nondecreasing order.



I = increasing (nondecreasing) order
D = decreasing (nonincreasing) order

Hypercube:

000	a_0	connected to 001, 010, and 100
001	a_1	connected to 000, 011, and 101
010	a_2	connected to 011, 000, and 110
011	a_3	connected to 010, 001, and 111
100	a_4	connected to 101, 110, and 000
101	a_5	connected to 100, 111, and 001
110	a_6	connected to 111, 100, and 010
111	a_7	connected to 110, 101, and 011

So, they are all connected to the nodes required by the C-I operations.

Note: need to be careful to always have a bitonic sequence to merge. This can be accomplished by being careful to sort into either ascending or descending order, as appropriate.

Finally, this gives a $\Theta(\log^2)$ time sorting algorithm for an *n*-processor hypercube or pram.