Sorting Networks

Overview

- Sorting Network Components
  - Comparison Networks
    - Comparator
  - Zero-one Principle
  - Bitonic Sorter
    - Half-Cleaner
    - Bitonic Sorter
  - Merging Network

- Bitonic Sorting Network
Comparison Networks

- Sorting networks are Comparison Networks that always sort their inputs

Comparison Network Components

- Wires
  - transmit values from one step to another
- Comparators
  - two inputs → two outputs (max and min)

Comparators

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>min(X, Y)</td>
</tr>
<tr>
<td>Y</td>
<td>max(X, Y)</td>
</tr>
</tbody>
</table>

\[X\rightarrow\text{comparator}\rightarrow\text{Y}\rightarrow\text{min}(X, Y)\rightarrow\text{max}(X, Y)\]
Comparison Networks

- **n** input wires: \( a_1, a_2, \ldots, a_n \)
  - input sequence: \( <a_1, a_2, \ldots, a_n> \)

- **n** output wires: \( b_1, b_2, \ldots, b_n \)
  - output sequence: \( <b_1, b_2, \ldots, b_n> \)

**Properties:**
- graph is acyclic
- output produced only when input is available
- comparators process in parallel if input is available

Comparison Network is like a procedure in that it specifies how comparisons are to occur

---

**Comparison Networks**

```
+---+---+---+---+
| A | C | 2 | 2 |
+---+---+---+---+
| 5 | 2 | 5 | 6 |
+---+---+---+---+
| 9 | 6 | 6 | 9 |
+---+---+---+---+
| a_1 | a_2 | a_3 | a_4 |
+---+---+---+---+

Depth
+---+---+---+---+
| 1 | 2 | 2 | 3 |
+---+---+---+---+
```

**Depth starts at 0**
- **input wires depth:** \( d_x \) and \( d_y \)
- **output wires depth:** \( \max(d_x, d_y) + 1 \)
### Sorting Networks

- A Comparison Network where the output is sorted
  - output sequence is *monotonically* increasing
  - \((b_1 \leq b_2 \ldots \leq b_n)\) for every input sequence

- Not all Comparison Networks are Sorting Networks

---

### Zero-One Principle

“If a Sorting Network works correctly when each input is drawn from the set \(\{0, 1\}\), then it works correctly on arbitrary input numbers”
**Zero-One Principle**

**Lemma 1**

If a Comparison Network transforms the input sequence \( a = <a_1, a_2, \ldots, a_n> \) into the output sequence \( b = <b_1, b_2, \ldots, b_n> \), then for any **monotonically** increasing function \( f \), the network transforms the input sequence:

\[
f(a) = <f(a_1), f(a_2), \ldots, f(a_n)>
\]

into the output sequence:

\[
f(b) = <f(b_1), f(b_2), \ldots, f(b_n)>
\]

**Monotonically Increasing Function:**

if \( a \geq b \) then \( f(a) \geq f(b) \),

---

**Proof:**

- if \( f \) is a monotonically increasing function, then a single comparator with inputs \( f(x) \) and \( f(y) \) produces output \( f(\min(x, y)) \) and \( f(\max(x, y)) \)

\[
\begin{align*}
  & f(x) \quad \min(f(x), f(y)) \\
  & f(y) \quad \max(f(x), f(y)) \\
\end{align*}
\]

\[
\begin{align*}
  \min(f(x), f(y)) &= f(\min(x, y)) \\
  \max(f(x), f(y)) &= f(\max(x, y))
\end{align*}
\]
Theorem 1: Zero-One Principle

If a Comparison Network with n inputs sorts all $2^n$ possible sequences of 0's and 1's correctly, then it sorts all sequences of arbitrary numbers correctly.
Zero-One Principle

Proof By Contradiction:
Suppose that the network sorts all zero-one sequences, but there exists a sequence of arbitrary numbers that the network does not sort:

\[ \langle a_1, a_2, \ldots, a_n \rangle \]
contains elements \( a_i \) and \( a_j \), where \( a_i < a_j \)

but the network places \( a_j \) before \( a_i \) in the output sequence: \( \langle \ldots, a_j, \ldots, a_i, \ldots \rangle \)

Define monotonically increasing function \( f \):

\[
f(x) = \begin{cases} 
0 & \text{if } x \leq a_i \\
1 & \text{if } x > a_i 
\end{cases}
\]

\( a_j \) is placed before \( a_i \) in the output sequence \( f(a_j) \) placed before \( f(a_i) \) when input is \( \langle a_1, a_2, \ldots, a_n \rangle \)
since \( f(a_j) = 1 \) and \( f(a_i) = 0 \), from above we have a contradiction
Bitonic Sequence

A sequence that monotonically increases and then decreases, or can be circularly shifted to become monotonically increasing and then monotonically decreasing.

Examples:
\(<1, 4, 6, 8, 3, 2>\) \(<6, 9, 4, 2, 3, 5>\)
\(<9, 8, 3, 2, 4, 6>\)

**Zero-one:** \(0^i0^j0^k\) or \(1^i0^j1^k\) when \(i,j,k \geq 0\)

Half Cleaner

- A Comparison Network of depth 1
- Line \(i\) compared with \(i + n/2\) for \(i = 1, 2, \ldots, n/2\)
- Assume that \(n\) is even
**Half Cleaner Example**

![Half Cleaner Example Diagram]

**Bitonic Sorting Network**

**Lemma 2**

If the input to a half-cleaner is a bitonic sequence of 0’s and 1’s, then the output satisfies the following properties:

- both the top half and the bottom half are bitonic
- every element in the top half is at least as small as every element in the bottom half
- at least one half is clean (either all 0’s or 1’s)
Proof

- The Comparison Network Half-Cleaner\[n\] compares inputs \(i\) and \(i + n/2\) for \(i = 1, 2, \ldots, n/2\)

- Suppose input is of the form: 
  00\ldots011\ldots100\ldots0

- There are 4 possible cases in total

Cases

- Bitonic
- Bitonic
- Bitonic
- Bitonic

GPU Programming, Shiraz University, Winter 88/Spring 89, Reza Azimi
Cases (continued)

Bitonic Sorter

- A sorting network that sorts bitonic sequences
- Built by recursively combining half-cleaners first
  - stage: Half-Cleaner[n]
  - subsequent sorts with Bitonic-Sorter[n/2]
- Depth of Bitonic-Sorter[n]: (shown as D(n))

\[
D(n) = \begin{cases} 
0 & \text{if } n = 1 \\
D(n/2) + 1 & \text{if } n = 2^k, k \geq 1 
\end{cases}
\]
Bitonic Sorter

Bitonic-Sorter[n]

Half-Cleaner[n]

Bitonic-Sorter[n/2]

Bitonic-Sorter[n/2]

---

Bitonic Sorting Network

**Example:**

<table>
<thead>
<tr>
<th>bitonic</th>
<th>sorted</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

---
Bitonic Sorting Network

- Bitonic-Sorter can be used to sort a bitonic zero-one sequence
- By the Zero-One principle it follows that any bitonic sequence of arbitrary numbers can be sorted using this network
- How about sorting non-bitonic sequences

A Merging Network

- Networks that can merge two sorted input sequences into one sorted output sequence
- The merging network is based on the following observation:
  - Given two sorted sequences, if we reverse the second sequence and then concatenate the two, the resulting sequence is bitonic.
  - Example
    - given: $X = 00001111$ and $Y = 00001111$
    - $Y^R = 11110000$
    - $X \cdot Y^R = 00001111111110000$
Merger

- We can construct \text{MERGER}[n] by modifying the first half cleaner of \text{BITONIC-SORTER}[n].

- The key is to perform the reversal of the second half of the inputs implicitly.

- Given two sorted sequences \( <a_1, a_2, \ldots, a_{n/2}> \) and \( <a_{n/2+1}, a_{n/2+2}, \ldots, a_n> \)

- Want the effect of bitonically sorting the sequence \( <a_1, a_2, \ldots, a_{n/2}, a_n, a_{n-1}, a_{n-2}, \ldots, a_{n/2+1}> \)

---

First Stage: Modifying Half-Cleaner

\[ \begin{align*}
\text{Sorted} & \quad 0 & \quad 0 \\
& 0 & 0 \\
& 1 & 1 \\
& 1 & 0 \\
& 0 & 1 \\
\text{Sorted} & \quad 1 & \quad 1 \\
& 1 & 1 \\
& 1 & 1 \\
& 1 & 1 \\
\end{align*} \]

\( \text{Bitonic} \)

\( \text{Bitonic Clean} \)
A Merging Network

- The reversal of a bitonic sequence is bitonic sequence.

- Both top and bottom outputs from the first stage of the merging network satisfy the properties of Lemma 2
  - both the top half and the bottom half are bitonic
  - every element in the top half is at least as small as every element in the bottom half
  - at least one half is clean (either all 0's or 1's)

The top and bottom outputs can be bitonically sorted in parallel to produce the sorted output of the merging network.
Merger

A Merging Network Example
A Merging Network

- The depth is the same as BITONIC-SORTER[n]

- Depth $D(n)$ of MERGING-NETWORK[n]:

$$D(n) = \begin{cases} 
0 & \text{if } n = 1 \\
D(n/2) + 1 & \text{if } n = 2^k, k \geq 1 
\end{cases}$$

$$D(n) = \log_2 N$$

A Sorting Network

- Now we have all the necessary tools to construct a network that can sort any input sequence

- We are going to use the aforementioned merging network to implement a parallel version of merge sort, called SORTER[n]
A Sorting Network

Sorter[n]

Sorter[n/2]  Merger[n]

Sorter[n/2]

A Sorting Network


### A Sorting Network

![Sorting Network Diagram]

### Sorting Network Depth

\[
D(n) = \begin{cases} 
0 & \text{if } n = 1 \\
D(n/2) + \log_2 N & \text{if } n = 2^k, k \geq 1 
\end{cases}
\]

\[
D(n) = (\log_2 N)^2
\]
Conclusion

- Sorting Network Components
  - Comparison Networks
  - Zero-one Principle
  - Bitonic Sorting Network
    - Half-Cleaner
    - Bitonic Sorter
  - Merging Network

- Sorting Networks