Recall from last time:
- Truncated TOM$(n, k)$ has area $O(n^2 k^2)$.
- TOM$(n)$ has area $O(n^2 l g^2 n)$.
- If $G$ has a $(w, \alpha)$ decom tree, it has a $(O(w), \alpha)$ balanced decom tree.

**Theorem**
Every $N$-node graph with a $(w, \sqrt{2})$ decom tree can be laid out in $O(w^2 l g^2 (N/w))$ area.

**Pf.**
Get an $(O(w), \sqrt{2})$ balanced decom tree for $G$.

Embed $G$ in TOM$(cw, 2 l g (N/w))$ for some constant $c$:

![Diagram](image)

**Leaf meshes:**
- #nodes $= \frac{(cw)^2}{2^{\frac{1}{2}l g (N/w)}} = \frac{c^2 w^2}{N^2 / w^2} = \frac{c^2 w^4}{N^2}$
- Side length $= \frac{c w^2}{N}$

- #vertices in leaf meshes $= \frac{N}{2^{\frac{1}{2}l g (N/w)}} = \frac{N}{N^2 / w^2} = \frac{w^2}{N}$

- #edges leaving leaf mesh $= \frac{O(w)}{(\sqrt{2})^{\frac{1}{2}l g (N/w)}} = \frac{O(w)}{N/w} = O\left(\frac{w^2}{N}\right)$

By adjusting $c$, can route edges within mesh + room on perimeter for wires to escape.
At depth $p$, side length of mesh is $\geq \frac{cw}{2^{p/2}}$.

#edges leaving = $O(w)/(\sqrt{2})^{p-1} = O(w)/2^{p/2}$.

Adjust $c$ for adequate capacity.

Routing internal-node meshes:

![Diagram of routing internal-node meshes]

Type 1 & 2: 2 layers each.
Type 3: 3 layers.

... 7 layers (squash to 2 if desired).

Area of $TOM(cw, 2\log(N/w))$ is

$O((cw)^2 (2\log(N/w))^2) = O(w^2 \log^2(N/w))$\(\checkmark\)

Corollary. Let $w$ be smallest value for a $(w, \sqrt{2})$ decomp tree for $N$-node graph $G$. Let $A$ be min area.

Then, $w^2 \leq A \leq O(w^2 \log^2(N/w))$.

PF.

![Diagram of area calculation]

#edges leaving subgraph at depth $p$

$\leq \frac{\sqrt{A}}{\sqrt{2}}^{p-1}$

$\therefore G$ has $(\sqrt{A}, \sqrt{2})$ decomp tree. $\Rightarrow w \leq \sqrt{A}$.\(\checkmark\)
\textbf{Which network is best?}

<table>
<thead>
<tr>
<th>Network</th>
<th>Area</th>
<th>Routing Time</th>
<th>$AT^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear array</td>
<td>$N$</td>
<td>$N$</td>
<td>$N^3$</td>
</tr>
<tr>
<td>2D array</td>
<td>$N$</td>
<td>$\sqrt{N}$</td>
<td>$N^2$</td>
</tr>
<tr>
<td>3D array</td>
<td>$N^{1/3}$</td>
<td>$N^{1/3}$</td>
<td>$N^2$</td>
</tr>
<tr>
<td>CBT</td>
<td>$N$</td>
<td>$N$</td>
<td>$N^3$</td>
</tr>
<tr>
<td>hypercube</td>
<td>$N^2$</td>
<td>$\lg N$</td>
<td>$N^2 \lg^2 N$</td>
</tr>
<tr>
<td>butterfly</td>
<td>$N^{2/3} N$</td>
<td>$\lg N$</td>
<td>$N^2$</td>
</tr>
<tr>
<td>2D MOT</td>
<td>$N \lg^2 N$</td>
<td>$\sqrt{N}$</td>
<td>$N^2 \lg^2 N$</td>
</tr>
</tbody>
</table>

**Universality:** An $N$-node butterfly can simulate any other $N$-node bounded-degree network with $O(\lg N)$ slowdown, just by routing messages.

Universal = expensive?

**VLSI perspective:** normalize to area, not \# procs.

Area $A$ network. Route $A$ packets:

- 2D array: $N = A$
  - Route $A$ packets in $\sqrt{A}$ time
- Butterfly: $N = \sqrt{A} \lg A$ ($\lg A \sim \lg N$)
  - Route $N$ packets in $\lg N$ time
    \[ \frac{A}{\sqrt{A} \lg A} = \frac{\sqrt{A}}{\lg A} \]

$TA/\lg A$ batches of $\sqrt{A} \lg A$ packets, each taking $\lg A$ time. Total time = $\sqrt{A} / \lg A \times \lg A = \sqrt{A}$.

Same! (Reason: basically since $AT^2 = N^2$ for both).

\textbf{How can we compare?} Ans. Simulate.

- Can an area-$A$ butterfly simulate any other area-$A$ network efficiently? ($O(\lg A)$ slowdown)
  - Can't even do linear array.

  \[ \# \text{procs in butterfly} = \sqrt{A} \lg A \]
  \[ \# \text{procs in lin. array} = A \]
  \[ \text{Slowdown} = \frac{A}{\sqrt{A} \lg A} = \frac{\sqrt{A}}{\lg A} \]
1D array can't sim. 2D array (draw)
2D array can't sim CBT (digraph).
CBT can't sim. 2D array (brs. width).
2D MOT can sim others with $\lg^2 A$ slowdown.

Next time: "Area-universal" networks.
   Idea: physical structure is TOM, but low dram.
   "Fat-trees"

«Reminder: catch up on reading for final>>