Contents of Networks part 2 Lecture

1. Embedding Networks
   1. Embedding a Linear Array onto a Hypercube
   2. Embedding a Mesh into a Hypercube
   3. Hypercube and Binary Search Trees
2. Butterfly Network

Broadcast on Hypercube

\[
\text{for } \text{Id} = 0 \text{ to } 2^d - 1 \text{ do in parallel}
\]
\[
\text{for stage} = 0 \text{ to } d \text{ do}
\]
\[
\text{partner} := \text{Id XOR } 2^{\text{stage}}
\]
\[
\text{if haveMessage is true then send message to partner}
\]
\[
\text{if partnerMessageQueue is not empty then read message}
\]
\end{for}
\end{for}
Scan on Hypercube

Assume each processor starts with a value stored in myNumber

for Id = 0 to \(2^d - 1\) do in parallel

    result := myNumber
    message := result

    for stage = 0 to d do
        partner := Id XOR \(2^{stage}\)
        send message to partner
        receive number from partner
        message := message + number
        if ( partner < Id ) then result := result + number
    end for
end for
Embedding Networks

Let \( G( V, E) \) and \( G'( V', E' ) \) be graphs

\( V, V' \) are set of vertices
\( E, E' \) are set of edges

A map \( H: G \rightarrow G' \) is an embedding if:

each vertex in \( V \) is mapped to a vertex in \( V' \)
and each edge in \( E \) is mapped to one or more edges in \( E' \)

Congestion

Maximum number of edges of \( E \) mapped on to any edge in \( E' \)

Dilation

Maximum number of edges of \( E' \) that any one edge of \( E \) is mapped onto
That is how much we stretch an edge of \( E \)

Expansion

\( |V'|/|V| \)

Load

Maximum number of vertices of \( E \) that are mapped to a single vertex of \( E' \)

Embedding a Linear Array onto a Hypercube

Embed a linear array of \( 2^d \) processors into \( d \)-dimensional Hypercube

1. Label the processors of the linear array in order from 0 to \( 2^d - 1 \)

2. Map linear array processor \( K \) to processor \( H( i, d) \) where:
\[ H(0,1) = 0 \]
\[ H(1,1) = 1 \]
\[ H(i,x+1) = \begin{cases} 
  H(i,x) & i < 2^x \\
  2^x + H(2^{x+1} - 1 - i, x) & i \geq 2^x 
\end{cases} \]

The sequence generated by the H's is a Gray code

---

**Gray Codes - The Easy Way**

1-bit code is 0 then 1

To get x-bit code:

- copy (x-1)-bit code, call it A
- repeat the (x-1)-bit code in reverse order, call it B
- add a 0 bit in front of the elements of A
- add a 1 bit in front of the elements of B

<table>
<thead>
<tr>
<th>1-bit</th>
<th>2-bit</th>
<th>3-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

---

**Sample Embedding**
Embedding a Mesh into a Hypercube

A $2^s \times 2^r$ mesh can be embedded into a s+r dimensional hypercube with dilation and congestion 1

proof:

1. Label the mesh using 2 dimensional coordinates

\[
\begin{array}{c c c c}
(0,0) & (0,1) & (0,2) & (0,3) \\
(1,0) & (1,1) & (1,2) & (1,3)
\end{array}
\]

2. Produce gray code for each dimension

<table>
<thead>
<tr>
<th>Number</th>
<th>Gray Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
</tr>
</tbody>
</table>
3. Concatenate gray codes to produce gray code for tuples
4. Gray code indicates the processor in the Hypercube the mesh processor will be mapped onto

<table>
<thead>
<tr>
<th>Tuple</th>
<th>Gray Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,0)</td>
<td>0 00</td>
</tr>
<tr>
<td>(0,1)</td>
<td>0 01</td>
</tr>
<tr>
<td>(0,2)</td>
<td>0 11</td>
</tr>
<tr>
<td>(0,3)</td>
<td>0 10</td>
</tr>
<tr>
<td>(1,0)</td>
<td>1 00</td>
</tr>
<tr>
<td>(1,1)</td>
<td>1 01</td>
</tr>
<tr>
<td>(1,2)</td>
<td>1 11</td>
</tr>
<tr>
<td>(1,3)</td>
<td>1 10</td>
</tr>
</tbody>
</table>

Theorem

A $2^{s_1} \times 2^{s_2} \times \ldots \times 2^{s_k}$ mesh can be embedded into a $s_1 + s_2 + \ldots + s_k$-dimensional hypercube with dilation and congestion 1.

Hypercube and Binary Search Trees

A complete binary search tree is a binary search tree such that:

All internal nodes have two children, with one possible exception.
All leaves occur on at most two different, but consecutive, levels
If a level contains leaves and internal nodes, the internal nodes must be to the left of all leaves, internal
nodes with two children must be to the left of internal nodes with one child

Theorem

A complete binary search tree with $2^r - 1$ nodes can not be embedded into a r-dimensional hypercube
with dilation and congestion 1 when $r > 2$

proof:

Assume you can and the root of the tree is mapped to node 0 in the hypercube
Label the nodes in the hypercube either even or odd, depending on the sum of the bits of the processor
number
There will be $2^{r-1}$ odd nodes and $2^{r-1}$ even nodes

Label the nodes in the tree even or odd, depending on which type of node it is mapped to in the hypercube
All nodes on the same level will have the same label

Lemma

A complete binary search tree with $2^r - 1$ nodes has $2^{r-1}$ leaves
proof:

Let $X$ be the polarity (odd or even) of the lowest level.

If there are more than 2 levels, there will be at least $2^{r-1} + 1$ nodes of polarity $X$.

But an $r$-dimensional hypercube does not have $2^{r-1} + 1$ nodes of the same polarity.

---

**Butterfly Network**

The $r$-dimensional butterfly has $(r+1)2^r$ nodes and $r2^{r+1}$ edges.

Processors (nodes) are labeled as $(w, i)$ where:

- $i$ denotes the level or dimension of the processor
- $0 \leq i \leq r$
- $w$ is an $r$-bit binary number that denotes the row of the processor

Two processors $(w, i)$ and $(w', i')$ are connected if and only if:

- $i' = i + 1$ and either
  - $w = w'$ (straight edge)
  - $w$ and $w'$ differ in the $i$'th bit (cross edge)

---

**Three-Dimensional Butterfly**
Butterfly and Hypercube

The hypercube is a folded up butterfly

The k'th row of the butterfly corresponds to the k'th processor in the hypercube

The k'th dimensional edge \((u, v)\) of the hypercube corresponds to the cross edges \((<u, k-1>, <v, k>)\) and \((<v, k-1>, <u, k>)\)