Embedding

- An embedding describes how the nodes of one (domain) network can be mapped into another (range) network.
  - Each node in range network is the target of at most one node in the domain network, unless specified otherwise.
  - Should keep the domain network as large as possible with respect to the range network.
  - Having the same number of nodes is perfect.
  - Each link of domain network maps onto a path in the range network.
  - Having a link map onto a link is perfect.
- A "perfect" embedding of a ring onto a torus is shown in Fig. 1.15.
- A "perfect" embedding of a mesh/torus in a hypercube is given in Figure 1.16.
- The *dilation* of an embedding is the maximum number of links in the range network corresponding to one link in the domain network.
  - Perfect embeddings have a dilation of 1.
- Embedding of binary trees in other networks are used in Ch. 3 -4 for performing broadcasts and reductions.

- Some results on binary trees embeddings follow.
  - *Theorem:* A complete binary tree of height greater than 4 can not be embedded in a 2-D mesh with a dilation of 1. (Quinn, 1994, pg135)
  - *Exercise*: A dilation-2 embedding of a binary tree of height 4 is shown in Fig. 1.17. Find a dilation-1 embedding of this binary tree.
  - *Theorem*: There exists an embedding of a complete binary tree of height n into a 2D mesh with dilation $\lceil n/2 \rceil$.
  - *Theorem*: A complete binary tree of height n has a dilation -2 embedding in a hypercube of dimension $n+1$ for all $n > 1$.
- **Note:** Network embeddings allow algorithms for the domain network to be transferred to the target nodes of the range network.

Communication Methods

- Two basic ways of transferring messages from source to destination.
  - **Circuit switching**
    - Establishing a path and allowing the entire message to transfer uninterrupted.
    - Similar to telephone connection that is held until the end of the call.
    - Links are not available to other messages until the transfer is complete.
    - Latency: If length of control packet sent to establish path is small wrt the message length, the the latency is essentially constant
      - *L/B,* where *L* is message length & *B* is bandwidth.
  - **packet switching**
    - Message is divided into "packets" of information
    - Each packet includes source and destination addresses.
    - Packets can not exceed a fixed, maximum size.
    - A packet is stored in a node in a buffer until it can move to the next node

Communications (cont)

    - Significant latency is created by storing each packet in each node it reaches.
    - Latency: increases linearly with the length of the route.
- *Virtual cut-through* package switching can be used to reduce the latency.
  - Allows packet to pass through a node without being stored, if the outgoing link is available.
  - If complete path is available, a message can immediately move from source to destination.
  - Latency: If flit is small, compared to message length, then the latency is essentially constant.
- *Wormhole Routing* alternate to store-and-forward packet routing
  - A message is divided into smaller units called *flits* (flow control units).
  - flits are 1-2 bytes in size.
  - can be transferred in parallel on links with multiple wires.
  - Only head of flit is initially transferred when the next link becomes available.
  - As each flit moves forward, the next flit can move forward.

Communications (cont)

- The entire path must be reserved for a message as these packets pull each other along (like cars of a train).
- Request/acknowledge bit messages are required to coordinate these pull-along moves. (See text)
- The complete path must be reserved, as these flits are linked together.
- Latency: If the head of the flit is very small compared to the length of the message, then the latency is essentially the constant L/B, with L the message length and B the link bandwidth.

- **Deadlock**
  - Routing algorithms needed to find a path between the nodes.
  - Adaptive routing algorithms choose different paths, depending on traffic conditions.
  - *Livelock* is a deadlock-type situation where a packet continues to go around the network, without ever reaching its destination.
  - *Deadlock*: No packet can be forwarded because they are blocked by other stored packets waiting to be forwarded.
- **Input/Output:** A significant problem on all parallel computers.

Metrics for Evaluating Parallelism

- Granularity:
  - MIMD computation requires that the task be divided into tasks or processes that can be executed simultaneously.
  - In *course grained* granularity, each process requires a large number of sequential instructions.
  - In *fine grained* granularity, only one or a few sequential instructions are required.
- Granularity (Another Approach)
  - Defn: Size of the computation between communication or synchronization points.
  - Increasing granularity, using this defn
    - reduces expensive communications
    - reduces costs of process creation
    - but reduces the nr of concurrent processes
- Speedup
  - A measure of the increase in running time due to parallelism.
  - Based on running times, $S(n) = t_s/t_p$, where
    - t_s is the execution time on a single processor, using the fastest known sequential algorithm and

Parallel Metrics (cont)

    - t_p is the execution time using a parallel processor.
  - In theoretical analysis, $S(n) = t_s/t_p$ where
    - t_s is the worst case running time for of the fastest known sequential algorithm for the problem
    - t_p is the worst case running time of the parallel algorithm using n PEs.
  - With traditional problems, the speedup of a parallel computer with n PEs is n and is called *linear speedup*. The argument is as follows::
    - Assume computation is divided perfectly into n processes of equal duration.
    - Assume no overhead is incurred
    - Then then optimal parallel running time of n is obtained
    - This yields an absolute maximal running time of t_s/n.
    - Then $S(n) = t_s/(t_s/n) = n$.
  - Normally, the speedup is much less than n, as
    - above assumptions usually do not occur.
    - Usually some parts of programs are sequential and only one PE is active

Parallel Metrics (cont)

    - During parts of the execution, some PEs are waiting for data to be received or to send messages.
- *Superlinear speedup* occurs if $S(n) > n$.
  - Textbook states that while this can happen, it is rare and due to reasons such as
    - extra memory in parallel system.
    - a suboptimal sequential algorithm used.
    - luck, in case of algorithm that has a random aspect in its design (e.g., quicksort)
  - Selim Akl has shown that for some less standard problems, superlinearity can be expected.
    - Some problems can not be solved without use of parallel computation.
    - Some problems are natural to solve using parallelism and sequential solutions are inefficient.
    - A whole chapter of his textbook and several journal papers have been written to defend these claims, but it may be a long time before they are accepted.
    - Superlinearity has had a hotly debated topic for some time.