CS252
Graduate Computer Architecture
Lecture 14

Multiprocessor Networks
March 9th, 2011

John Kubiatowicz
Electrical Engineering and Computer Sciences
University of California, Berkeley

http://www.eecs.berkeley.edu/~kubitron/cs252

What is *Parallel* Architecture?

- A parallel computer is a collection of processing elements that cooperate to solve large problems
  - Most important new element: It is all about communication!
- What does the programmer (or OS or Compiler writer) think about?
  - Models of computation:
    - » PRAM? BSP? Sequential Consistency?
  - Resource Allocation:
    - » how powerful are the elements?
    - » how much memory?
- What mechanisms must be in hardware vs software
  - What does a single processor look like?
    - » High performance general purpose processor
    - » SIMD processor/Vector Processor
  - Data access, Communication and Synchronization
    - » how do the elements cooperate and communicate?
    - » how are data transmitted between processors?
    - » what are the abstractions and primitives for cooperation?

Parallel Programming Models

- Programming model is made up of the languages and libraries that create an abstract view of the machine
  - Shared Memory –
    - » different processors share a global view of memory
    - » may be cache coherent or not
    - » Communication occurs implicitly via loads and store
  - Message Passing –
    - » No global view of memory (at least not in hardware)
    - » Communication occurs explicitly via messages
- Data
  - What data is private vs. shared?
  - How is logically shared data accessed or communicated?
- Synchronization
  - What operations can be used to coordinate parallelism
  - What are the atomic (indivisible) operations?
- Cost
  - How do we account for the cost of each of the above?

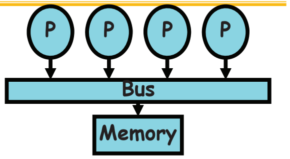
Flynn's Classification (1966)

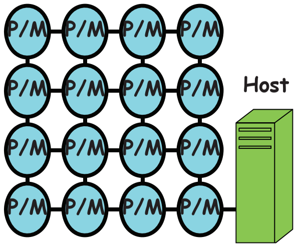
Broad classification of parallel computing systems

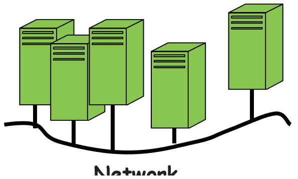
- SISD: Single Instruction, Single Data
  - conventional uniprocessor
- SIMD: Single Instruction, Multiple Data
  - one instruction stream, multiple data paths
  - distributed memory SIMD (MPP, DAP, CM-1&2, Maspar)
  - shared memory SIMD (STARAN, vector computers)
- MIMD: Multiple Instruction, Multiple Data
  - message passing machines (Transputers, nCube, CM-5)
  - non-cache-coherent shared memory machines (BBN Butterfly, T3D)
  - cache-coherent shared memory machines (Sequent, Sun Starfire, SGI Origin)
- MISD: Multiple Instruction, Single Data
  - Not a practical configuration

Examples of MIMD Machines

- Symmetric Multiprocessor
  - Multiple processors in box with shared memory communication
  - Current MultiCore chips like this
  - Every processor runs copy of OS
- Non-uniform shared-memory with separate I/O through host
  - Multiple processors
    - » Each with local memory
    - » general scalable network
  - Extremely light "OS" on node provides simple services
    - » Scheduling/synchronization
  - Network-accessible host for I/O
- Cluster
  - Many independent machine connected with general network
  - Communication through messages







Paper Discussion: "Future of Wires"

- "Future of Wires," Ron Ho, Kenneth Mai, Mark Horowitz
- Fanout of 4 metric (FO4)
  - FO4 delay metric across technologies roughly constant
  - Treats 8 FO4 as absolute minimum (really says 16 more reasonable)
- Wire delay
  - Unbuffered delay: scales with $(\text{length})^2$
  - Buffered delay (with repeaters) scales closer to linear with length
- Sources of wire noise
  - Capacitive coupling with other wires: Close wires
  - Inductive coupling with other wires: Can be far wires

"Future of Wires" continued

- Cannot reach across chip in one clock cycle!
  - This problem increases as technology scales
  - Multi-cycle long wires!
- Not really a wire problem – more of a CAD problem??
  - How to manage increased complexity is the issue
- Seems to favor ManyCore chip design??

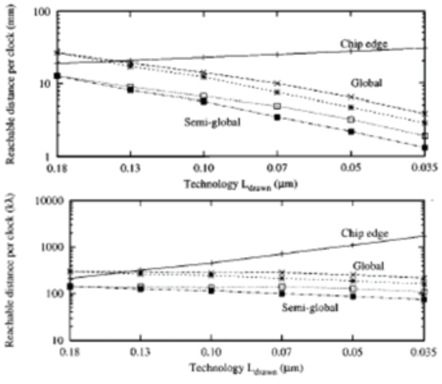


Fig. 20. Reachable distance per clock, repeated global wires.

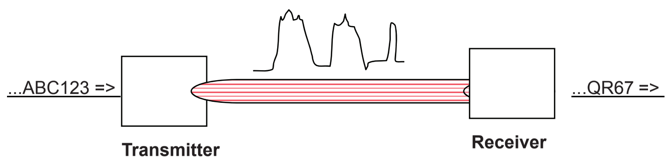
What characterizes a network?

- Topology (what)
  - physical interconnection structure of the network graph
  - direct: node connected to every switch
  - indirect: nodes connected to specific subset of switches
- Routing Algorithm (which)
  - restricts the set of paths that msgs may follow
  - many algorithms with different properties
    - » deadlock avoidance?
- Switching Strategy (how)
  - how data in a msg traverses a route
  - circuit switching vs. packet switching
- Flow Control Mechanism (when)
  - when a msg or portions of it traverse a route
  - what happens when traffic is encountered?

Formalism

- network is a graph V = {switches and nodes} connected by communication channels $C \subseteq V \times V$
- Channel has width w and signaling rate $f = 1/\tau$
  - channel bandwidth $b = wf$
  - phit (physical unit) data transferred per cycle
  - flit - basic unit of flow-control
- Number of input (output) channels is switch *degree*
- Sequence of switches and links followed by a message is a *route*
- Think streets and intersections

Links and Channels

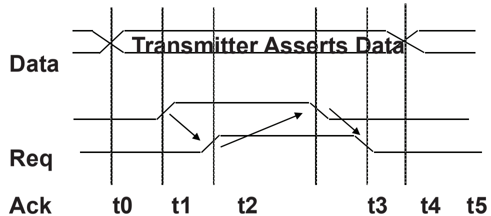


- *transmitter* converts stream of digital symbols into signal that is driven down the link
- *receiver* converts it back
  - tran/rcv share *physical protocol*
- trans + link + rcv form *Channel* for digital info flow between switches
- *link-level protocol* segments stream of symbols into larger units: packets or messages (framing)
- *node-level protocol* embeds commands for dest communication assist within packet

---

Clock Synchronization?

- Receiver must be synchronized to transmitter
  - To know when to latch data
- Fully Synchronous
  - Same clock and phase: Isochronous
  - Same clock, different phase: Mesochronous
    - » High-speed serial links work this way
    - » Use of encoding (8B/10B) to ensure sufficient high-frequency component for clock recovery
- Fully Asynchronous
  - No clock: Request/Ack signals
  - Different clock: Need some sort of clock recovery?



Administrative

- Exam: This Wednesday (3/30)
  Location: TBA
  TIME: TBA
  - This info is on the Lecture page (has been)
  - Get on 8 ½ by 11 sheet of notes (both sides)
  - Meet at LaVal's afterwards for Pizza and Beverages
- Assume that major papers we have discussed may show up on exam

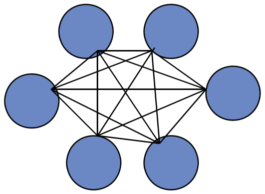
Topological Properties

- *Routing Distance* - number of links on route
- *Diameter* - maximum routing distance
- *Average Distance*
- A network is *partitioned* by a set of links if their removal disconnects the graph

---

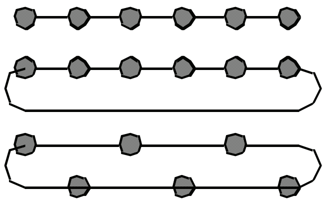
Interconnection Topologies

- Class of networks scaling with N
- Logical Properties:
  - distance, degree
- Physical properties
  - length, width



- Fully connected network
  - diameter = 1
  - degree = N
  - cost?
    - » bus => O(N), but BW is O(1) - actually worse
    - » crossbar => $O(N^2)$ for BW O(N)
- VLSI technology determines switch degree

Example: Linear Arrays and Rings



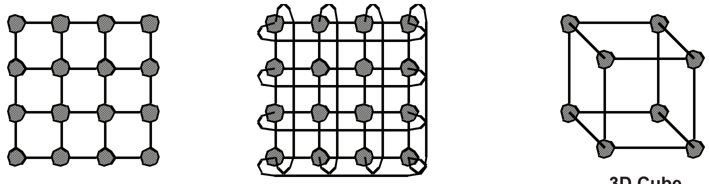
Linear Array

Torus

Torus arranged to use short wires

- Linear Array
  - Diameter?
  - Average Distance?
  - Bisection bandwidth?
  - Route A -> B given by relative address R = B-A
- Torus?
- Examples: FDDI, SCI, FiberChannel Arbitrated Loop, KSR1

Example: Multidimensional Meshes and Tori



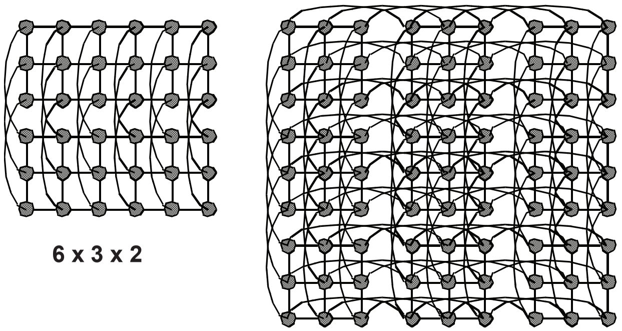
2D Grid

2D Torus

3D Cube

- *n*-dimensional array
  - $N = k_{n-1} \text{ X } ... \text{X } k_O$ nodes
  - described by *n*-vector of coordinates $(i_{n-1}, ..., i_O)$
- *n*-dimensional *k*-ary mesh: $N = k^n$
  - $k = \sqrt[n]{N}$
  - described by *n*-vector of radix k coordinate
- *n*-dimensional *k*-ary torus (or *k*-ary *n*-cube)?

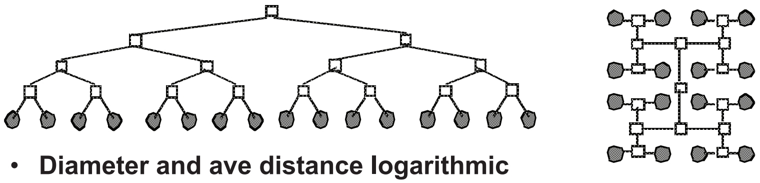
On Chip: Embeddings in two dimensions



- Embed multiple logical dimension in one physical dimension using long wires
- When embedding higher-dimension in lower one, either some wires longer than others, or all wires long

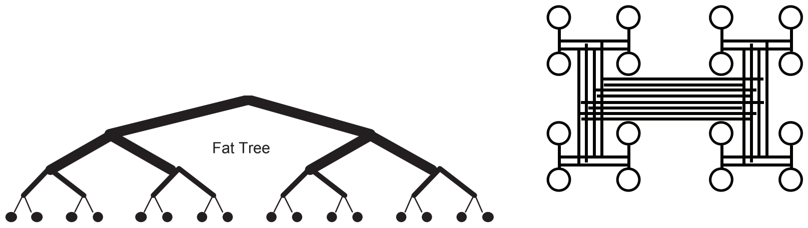
---

Trees



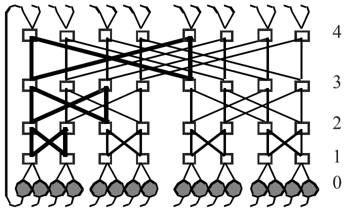
- Diameter and ave distance logarithmic
  - k-ary tree, height n = $\log_k N$
  - address specified n-vector of radix k coordinates describing path down from root
- Fixed degree
- Route up to common ancestor and down
  - R = B xor A
  - let i be position of most significant 1 in R, route up i+1 levels
  - down in direction given by low i+1 bits of B
- H-tree space is O(N) with O(√N) long wires
- Bisection BW?

Fat-Trees

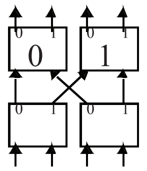


- Fatter links (really more of them) as you go up, so bisection BW scales with N

Butterflies



16 node butterfly



building block

- Tree with lots of roots!
- N log N (actually N/2 x logN)
- Exactly one route from any source to any dest
- R = A xor B, at level i use 'straight' edge if r_i=0, otherwise cross edge
- Bisection N/2 vs N $^{(n-1)/n}$ (for n-cube)

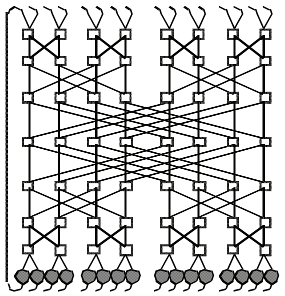
k-ary n-cubes vs k-ary n-flies

- degree n	vs degree k
- N switches	vs N log N switches
- diminishing BW per node	vs constant
- requires locality	vs little benefit to locality

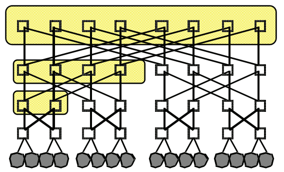
- Can you route all permutations?

Benes network and Fat Tree

16-node Benes Network (Unidirectional)



16-node 2-ary Fat-Tree (Bidirectional)



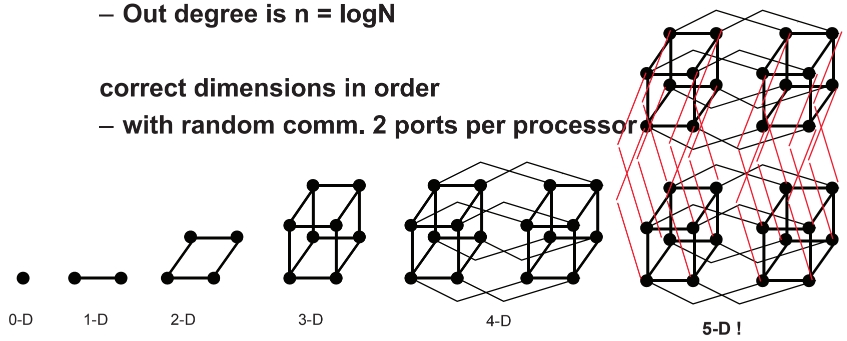
- Back-to-back butterfly can route all permutations
- What if you just pick a random mid point?

Hypercubes

- Also called binary n-cubes. # of nodes = N = 2^n.
- O(logN) Hops
- Good bisection BW
- Complexity
  - Out degree is n = logN

correct dimensions in order

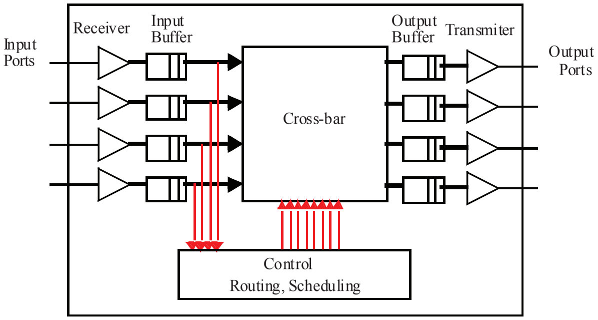
- with random comm. 2 ports per processor



Some Properties

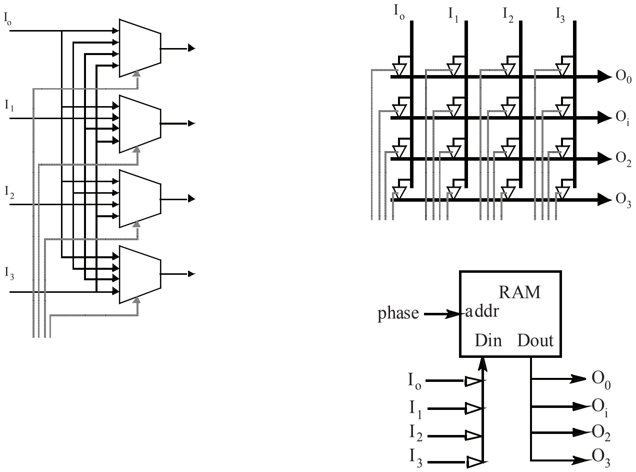
- Routing
  - relative distance: R = ($b_{n-1} - a_{n-1}$, ... , $b_0 - a_0$)
  - traverse ri = $b_i - a_i$ hops in each dimension
  - *dimension-order routing? Adaptive routing?*
- Average Distance — Wire Length?
  - n x 2k/3 for mesh
  - nk/2 for cube
- Degree?
- Bisection bandwidth? — Partitioning?
  - k^{n-1} bidirectional links
- Physical layout?
  - 2D in O(N) space — Short wires
  - higher dimension?

The Routing problem: Local decisions

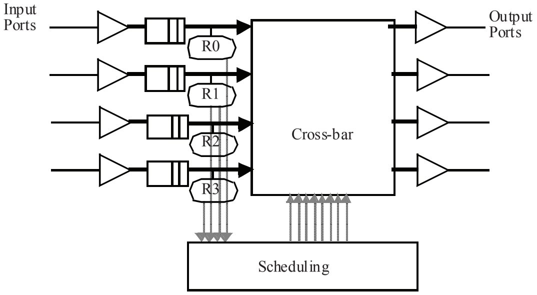

Input Ports
Receiver
Input Buffer
Cross-bar
Output Buffer
Transmiter
Output Ports
Control
Routing, Scheduling

- Routing at each hop: Pick next output port!

How do you build a crossbar?

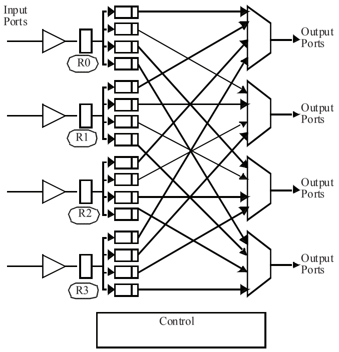

I_o
I_1
I_2
I_3
O_0
O_1
O_2
O_3
RAM
phase
addr
Din
Dout

Input buffered switch


Input Ports
Output Ports
R0
R1
R2
R3
Cross-bar
Scheduling

- Independent routing logic per input
  - FSM
- Scheduler logic arbitrates each output
  - priority, FIFO, random
- Head-of-line blocking problem
  - Message at head of queue blocks messages behind it

Output Buffered Switch


Input Ports
Output Ports
Output Ports
Output Ports
Output Ports
R0
R1
R2
R3
Control

- How would you build a shared pool?

Summary #1

- Network Topologies:

Topology	Degree	Diameter	Ave Dist	Bisection	D (D ave) @ P=1024
1D Array	2	N-1	N / 3	1	huge
1D Ring	2	N/2	N/4	2	
2D Mesh	4	2 ($N^{1/2}$ - 1)	2/3 $N^{1/2}$	$N^{1/2}$	63 (21)
2D Torus	4	$N^{1/2}$	1/2 $N^{1/2}$	$2N^{1/2}$	32 (16)
k-ary n-cube	2n	nk/2	nk/4	nk/4	15 (7.5) @n=3
Hypercube	n =log N		n	n/2	N/2 10 (5)

- Fair metrics of comparison
  - Equal cost: area, bisection bandwidth, etc

Summary #2

- Routing Algorithms restrict the set of routes within the topology
  - simple mechanism selects turn at each hop
  - arithmetic, selection, lookup
- Virtual Channels
  - Adds complexity to router
  - Can be used for performance
  - Can be used for deadlock avoidance
- Deadlock-free if channel dependence graph is acyclic
  - limit turns to eliminate dependences
  - add separate channel resources to break dependences
  - combination of topology, algorithm, and switch design
- Deterministic vs adaptive routing