CS252
Graduate Computer Architecture
Lecture 14

Multiprocessor Networks
March 7th, 2012

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

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

Review: VLIW: Very Large Instruction Word

- Each "instruction" has explicit coding for multiple operations
  - In IA-64, grouping called a "packet"
  - In Transmeta, grouping called a "molecule" (with "atoms" as ops)
- Tradeoff instruction space for simple decoding
  - The long instruction word has room for many operations
  - By definition, all the operations the compiler puts in the long instruction word are independent => execute in parallel
  - E.g., 2 integer operations, 2 FP ops, 2 Memory refs, 1 branch
    - » 16 to 24 bits per field => 7*16 or 112 bits to 7*24 or 168 bits wide
  - Need compiling technique that schedules across several branches

Problems with 1st Generation VLIW

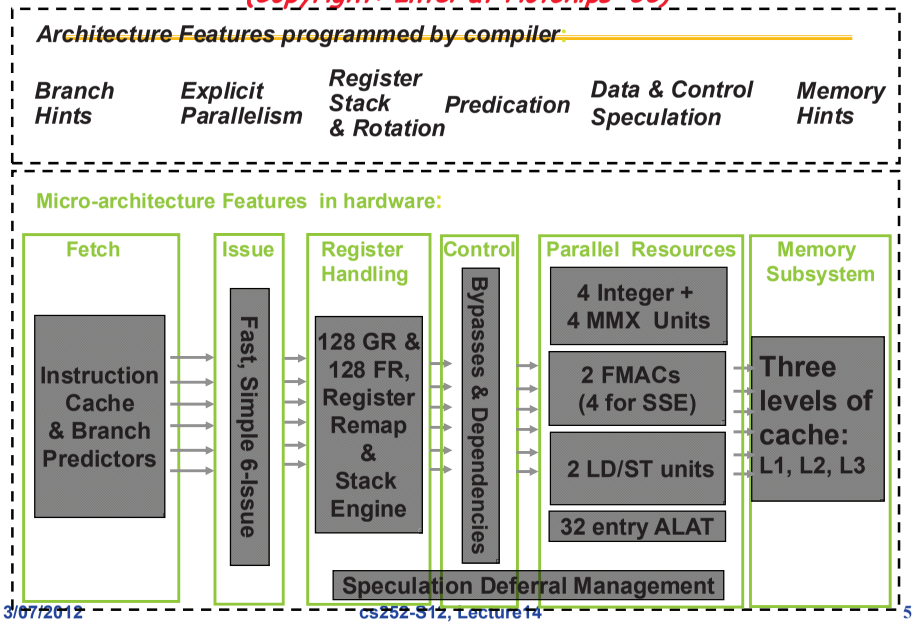
- Increase in code size
  - generating enough operations in a straight-line code fragment requires ambitiously unrolling loops
  - whenever VLIW instructions are not full, unused functional units translate to wasted bits in instruction encoding
- Operated in lock-step; no hazard detection HW
  - a stall in any functional unit pipeline caused entire processor to stall, since all functional units must be kept synchronized
  - Compiler might prediction function units, but caches hard to predict
- Binary code compatibility
  - Pure VLIW => different numbers of functional units and unit latencies require different versions of the code

Intel/HP IA-64 "Explicitly Parallel Instruction Computer (EPIC)"

- IA-64: instruction set architecture
  - 128 64-bit integer regs + 128 82-bit floating point regs
    - » Not separate register files per functional unit as in old VLIW
  - Hardware checks dependencies (interlocks ⇒ binary compatibility over time)
- 3 Instructions in 128 bit "bundles"; field determines if instructions dependent or independent
  - Smaller code size than old VLIW, larger than x86/RISC
  - Groups can be linked to show independence > 3 instr
- Predicated execution (select 1 out of 64 1-bit flags) ⇒ 40% fewer mispredictions?
- Speculation Support:
  - deferred exception handling with "poison bits"
  - Speculative movement of loads above stores + check to see if incorect
- Itanium™ was first implementation (2001)
  - Highly parallel and deeply pipelined hardware at 800Mhz
  - 6-wide, 10-stage pipeline at 800Mhz on 0.18 µ process
- Itanium 2™ is name of 2nd implementation (2005)
  - 6-wide, 8-stage pipeline at 1666Mhz on 0.13 µ process
  - Caches: 32 KB I, 32 KB D, 128 KB L2I, 128 KB L2D, 9216 KB L3

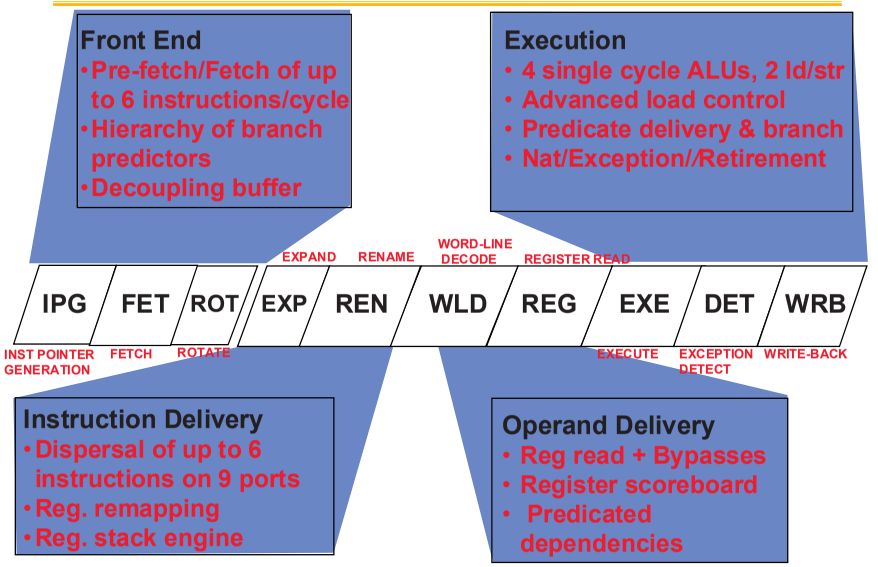
Itanium™ EPIC Design Maximizes SW-HW Synergy

(Copyright: Intel at Hotchips '00)



10 Stage In-Order Core Pipeline

(Copyright: Intel at Hotchips '00)



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?

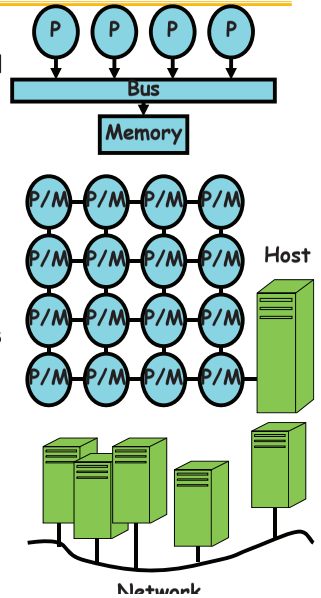
Flynn's Classification (1966)

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 (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??

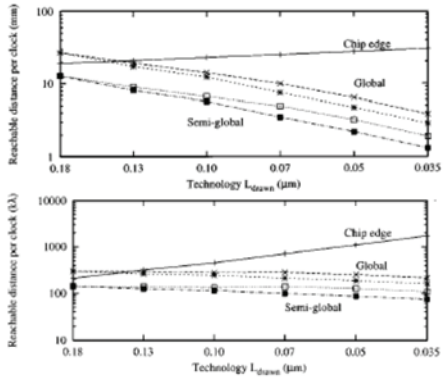


Fig. 20. Reachable distance per clock, repeated global wires.

What characterizes a network?

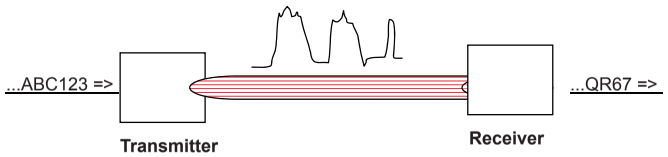
- Topology (what)
  - physical interconnection structure of the network graph
  - direct: node connected to every switch
  - indirect: nodes connected to specific subset of switches
- Routing Algorithm (which)
  - restricts the set of paths that msgs may follow
  - many algorithms with different properties
    - » 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

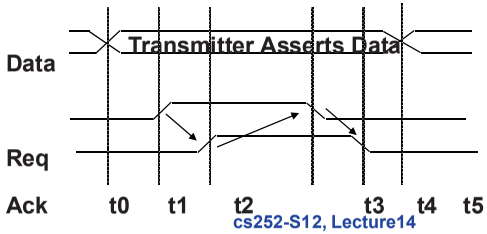
Links and Channels



- *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

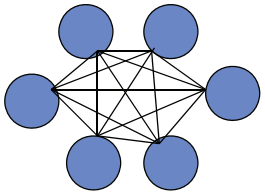
- Exam: Two Weeks from Today (3/21)
  Location: 405 Soda Hall
  TIME: 5:00—8:00
  - This info is on the Lecture page (has been)
  - Get on 8 ½ by 11 sheet of notes (both sides)
  - Meet at LaVal's afterwards for Pizza and Beverages
  - Bring dumb calculator (no network connection)
- Assume that major 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

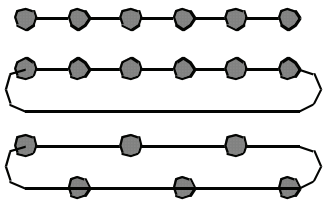
Interconnection Topologies

- 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



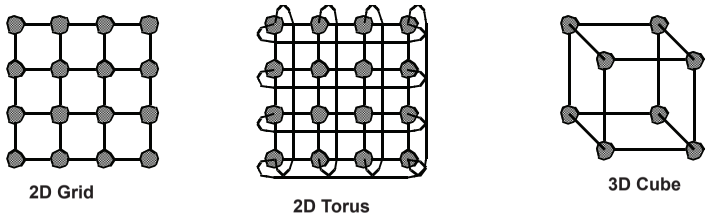
Linear Array

Torus

Torus arranged to use short wires

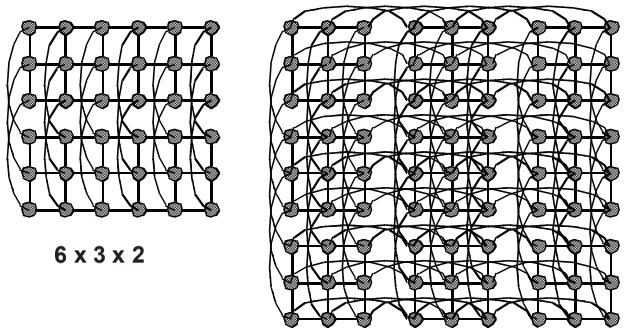
- Linear Array
  - Diameter?
  - Average Distance?
  - Bisection bandwidth?
  - Route A -> B given by relative address R = B-A
- Torus?
- Examples: FDDI, SCI, FiberChannel Arbitrated Loop, KSR1

Example: Multidimensional Meshes and Tori



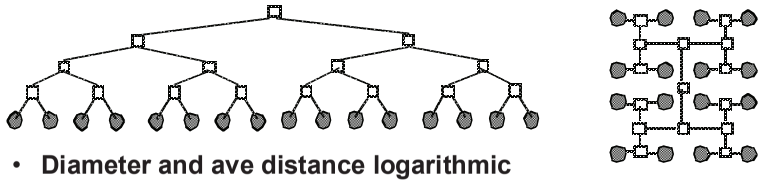
- *n*-dimensional array
  - N = k_{n-1} X ...X k_O nodes
  - described by *n*-vector of coordinates (i_{n-1}, ..., i_O)
- *n*-dimensional *k*-ary mesh: N = k^n
  - k = $\sqrt[n]{N}$
  - described by *n*-vector of radix k coordinate
- *n*-dimensional *k*-ary torus (or *k*-ary *n*-cube)?

On Chip: Embeddings in two dimensions



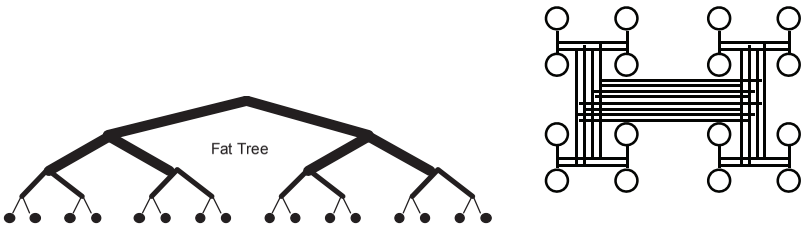
- Embed multiple logical dimension in one physical dimension using long wires
- When embedding higher-dimension in lower one, either some wires longer than others, or all wires long

Trees



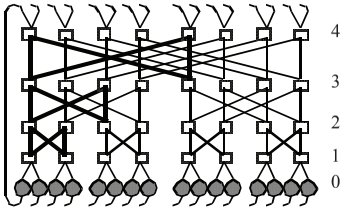
- Diameter and ave distance logarithmic
  - k-ary tree, height n = $\log_k N$
  - address specified n-vector of radix k coordinates describing path down from root
- Fixed degree
- Route up to common ancestor and down
  - R = B xor A
  - let i be position of most significant 1 in R, route up i+1 levels
  - 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

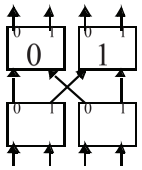


- Fatter links (really more of them) as you go up, so bisection BW scales with N

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)

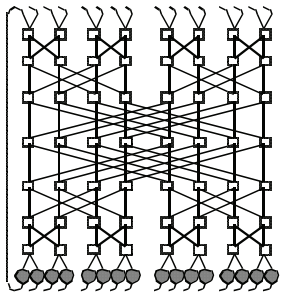
k-ary n-cubes vs k-ary n-flies

- 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

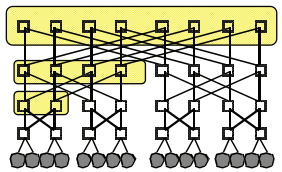
- Can you route all permutations?

Benes network and Fat Tree

16-node Benes Network (Unidirectional)



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

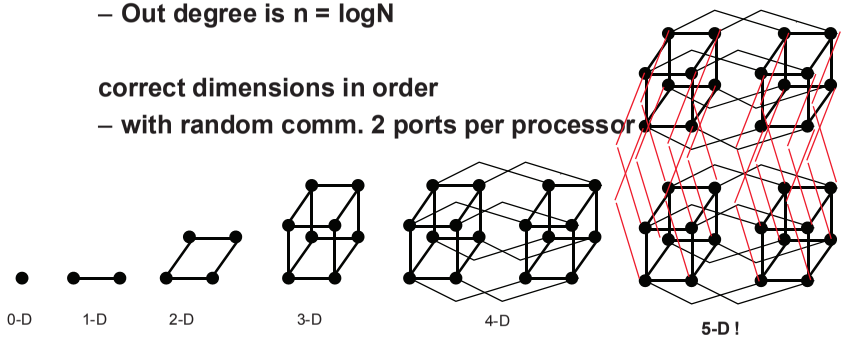


- Back-to-back butterfly can route all permutations
- What if you just pick a random mid point?

Hypercubes

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

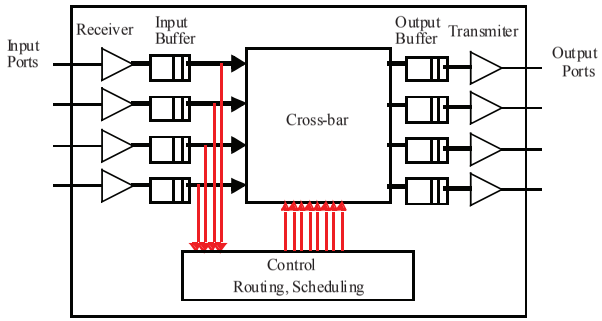
  correct dimensions in order
  - with random comm. 2 ports per processor



Some Properties

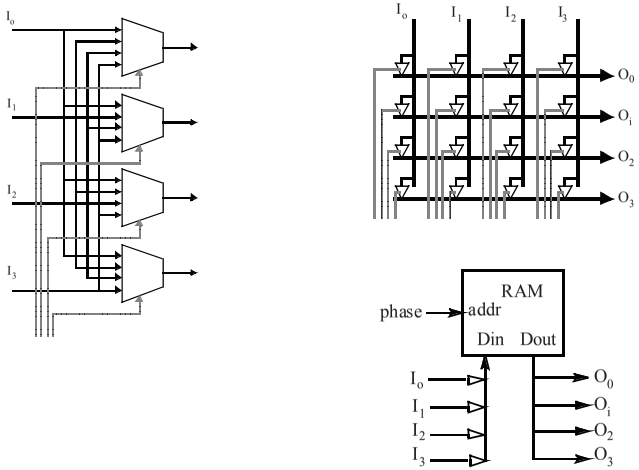
- Routing
  - relative distance: R = ($b_{n-1} - a_{n-1}$, ... , $b_0 - a_0$)
  - traverse ri = $b_i - a_i$ hops in each dimension
  - *dimension-order routing? Adaptive routing?*
- Average Distance — Wire Length?
  - n x 2k/3 for mesh
  - nk/2 for cube
- Degree?
- Bisection bandwidth? — Partitioning?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

The Routing problem: Local decisions

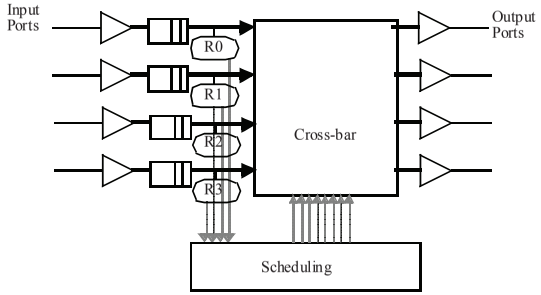


- Routing at each hop: Pick next output port!

How do you build a crossbar?

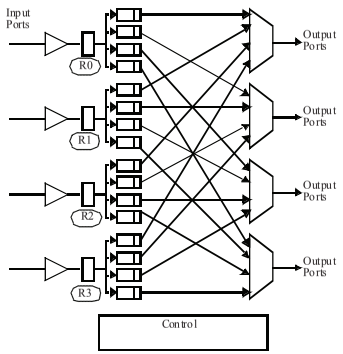


Input buffered switch



- 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



- How would you build a shared pool?

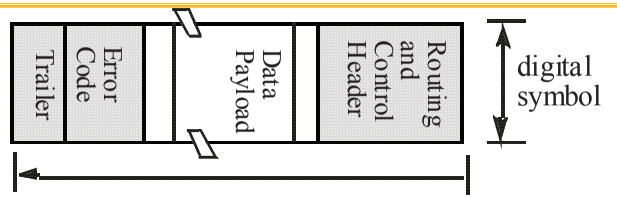
Properties of Routing Algorithms

- Routing algorithm:
  - R: N x N -> C, which at each switch maps the destination node n_d to the next channel on the route
  - which of the possible paths are used as routes?
  - how is the next hop determined?
    - » arithmetic
    - » source-based port select
    - » table driven
    - » general computation
- Deterministic
  - route determined by (source, dest), not intermediate state (i.e. traffic)
- Adaptive
  - route influenced by traffic along the way
- Minimal
  - only selects shortest paths
- Deadlock free
  - no traffic pattern can lead to a situation where packets are deadlocked and never move forward

Example: Simple Routing Mechanism

- need to select output port for each input packet
  - in a few cycles
- Simple arithmetic in regular topologies
  - ex: Δx, Δy routing in a grid
    - » west (-x) $\Delta x < 0$
    - » east (+x) $\Delta x > 0$
    - » south (-y) $\Delta x = 0, \Delta y < 0$
    - » north (+y) $\Delta x = 0, \Delta y > 0$
    - » processor $\Delta x = 0, \Delta y = 0$
- Reduce relative address of each dimension in order
  - Dimension-order routing in k-ary d-cubes
  - e-cube routing in n-cube

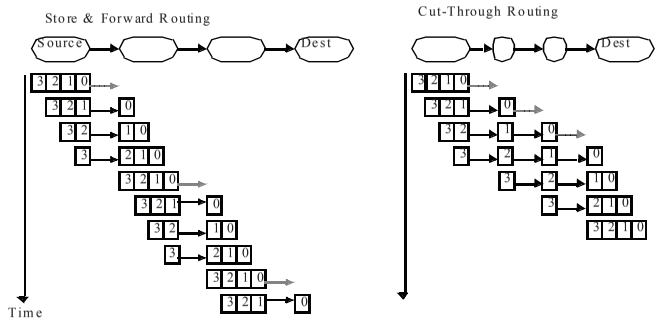
Communication Performance



Sequence of symbols transmitted over a channel

- Typical Packet includes data + encapsulation bytes
  - Unfragmented packet size $S = S_{data}+S_{encapsulation}$
- Routing Time:
  - $Time(S)_{s-d}$ = overhead + routing delay + channel occupancy + contention delay
  - Channel occupancy = $S/b = (S_{data}+ S_{encapsulation})/b$
  - Routing delay in cycles (Δ):
    - » Time to get head of packet to next hop
  - Contention?

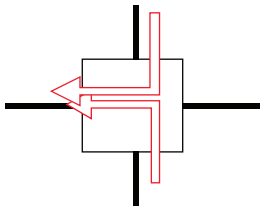
Store&Forward vs Cut-Through Routing



Time:	$h(S/b + \Delta/\tau)$	vs	$S/b + h\,\Delta/\tau$
OR(cycles):	$h(S/w + \Delta)$	vs	$S/w + h\,\Delta$

- what if message is fragmented?
- wormhole vs virtual cut-through

Contention

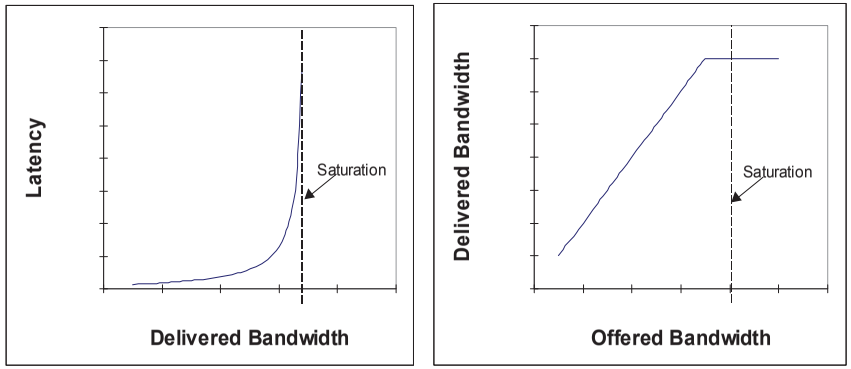


- Two packets trying to use the same link at same time
  - limited buffering
  - drop?
- Most parallel mach. networks block in place
  - link-level flow control
  - tree saturation
- Closed system - offered load depends on delivered
  - Source Squelching

Bandwidth

- What affects local bandwidth?
  - packet density: $b \times S_{data}/S$
  - routing delay: $b \times S_{data}/(S + w\Delta)$
  - contention
    - » endpoints
    - » within the network
- Aggregate bandwidth
  - bisection bandwidth
    - » sum of bandwidth of smallest set of links that partition the network
  - total bandwidth of all the channels: Cb
  - suppose N hosts issue packet every M cycles with ave dist
    - » each msg occupies h channels for l = S/w cycles each
    - » C/N channels available per node
    - » link utilization for store-and-forward: $\rho = (hl/M \text{ channel cycles/node})/(C/N) = Nhl/MC < 1$!
    - » link utilization for wormhole routing?

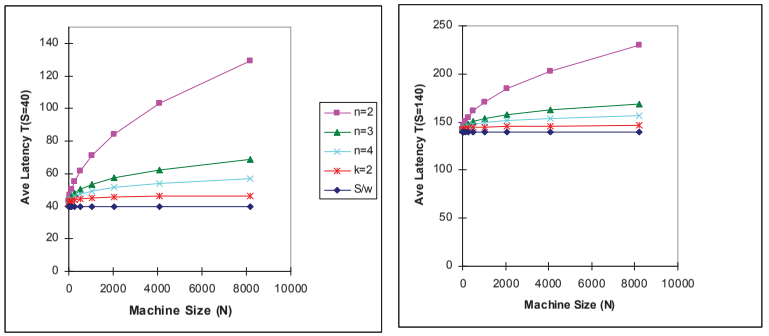
Saturation



How Many Dimensions?

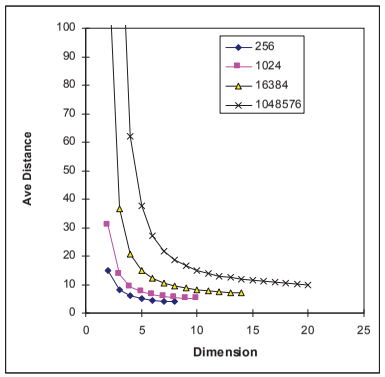
- n = 2 or n = 3
  - Short wires, easy to build
  - Many hops, low bisection bandwidth
  - Requires traffic locality
- n >= 4
  - Harder to build, more wires, longer average length
  - Fewer hops, better bisection bandwidth
  - Can handle non-local traffic
- k-ary n-cubes provide a consistent framework for comparison
  - $N = k^n$
  - scale dimension (n) or nodes per dimension (k)
  - assume cut-through

Traditional Scaling: Latency scaling with N



- Assumes equal channel width
  - independent of node count or dimension
  - dominated by average distance

Average Distance

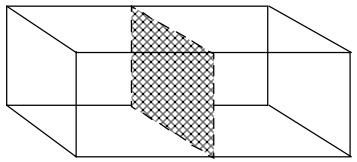


ave dist = n(k-1)/2

- but, equal channel width is not equal cost!
- Higher dimension => more channels

Dally Paper: In the 3D world

- For N nodes, bisection area is $O(N^{2/3})$



- For large N, bisection bandwidth is limited to $O(N^{2/3})$
  - Bill Dally, IEEE TPDS, [Dal90a]
  - For fixed bisection bandwidth, low-dimensional k-ary n-cubes are better (otherwise higher is better)
  - i.e., a few short fat wires are better than many long thin wires
  - What about many long fat wires?

Dally paper (con't)

- Equal Bisection,W=1 for hypercube ⇒ W= ½k
- Three wire models:
  - Constant delay, independent of length
  - Logarithmic delay with length (exponential driver tree)
  - Linear delay (speed of light/optimal repeaters)

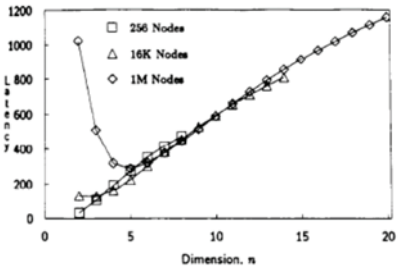


Fig. 9. Latency versus dimension for 256, 16K, and 1M nodes, logarithmic delay.

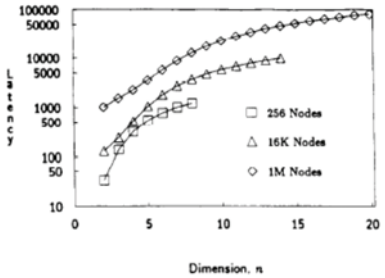


Fig. 10. Latency versus dimension for 256, 16K, and 1M nodes, linear delay.

Logarithmic Delay

Linear Delay

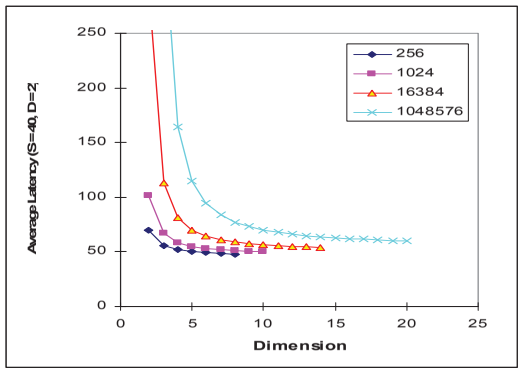
Equal cost in k-ary n-cubes

- Equal number of nodes?
- Equal number of pins/wires?
- Equal bisection bandwidth?
- Equal area?
- Equal wire length?

What do we know?

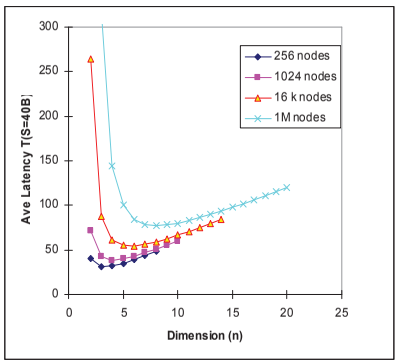
- switch degree: n diameter = n(k-1)
- total links = Nn
- pins per node = 2wn
- bisection = k^{n-1} = N/k links in each directions
- 2Nw/k wires cross the middle

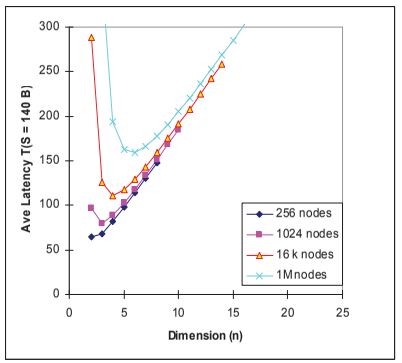
Latency for Equal Width Channels



- total links(N) = Nn

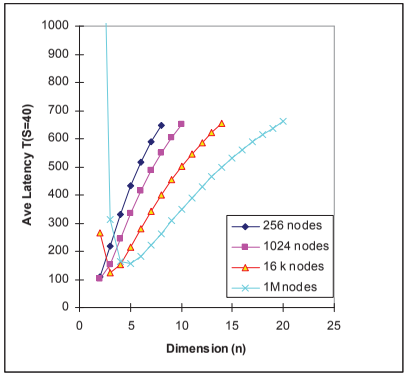
Latency with Equal Pin Count





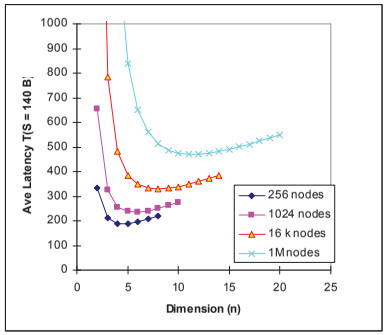
- Baseline n=2, has w = 32 (128 wires per node)
- fix 2nw pins => w(n) = 64/n
- distance up with n, but channel time down

Latency with Equal Bisection Width



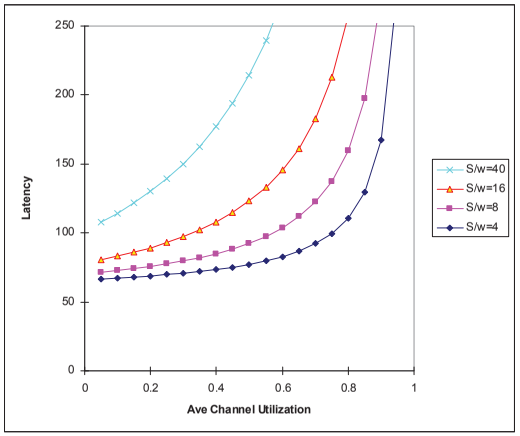
- N-node hypercube has N bisection links
- 2d torus has $2N^{1/2}$
- Fixed bisection $\Rightarrow$ $w(n) = N^{1/n} / 2 = k/2$
- 1 M nodes, n=2 has w=512!

Larger Routing Delay (w/ equal pin)



- Dally's conclusions strongly influenced by assumption of small routing delay
  - Here, Routing delay Δ=20

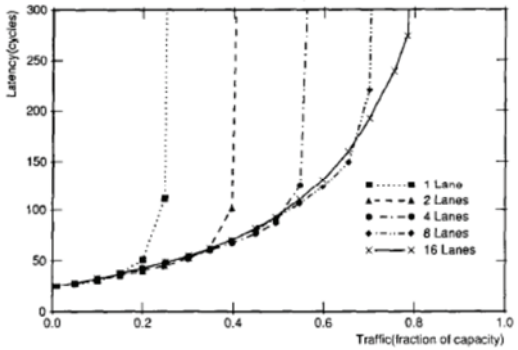
Saturation



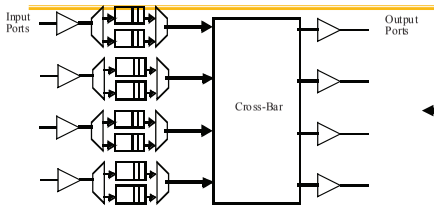
- Fatter links shorten queuing delays

Discuss of paper: Virtual Channel Flow Control

- Basic Idea: Use of virtual channels to reduce contention
  - Provided a model of k-ary, n-flies
  - Also provided simulation
- Tradeoff: Better to split buffers into virtual channels
  - Example (constant total storage for 2-ary 8-fly):



When are virtual channels allocated?

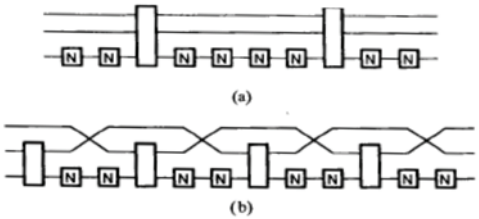


Hardware efficient design For crossbar

- Two *separate processes:*
  - Virtual channel allocation
  - Switch/connection allocation
- Virtual Channel Allocation
  - Choose route and free output virtual channel
  - Really means: Source of link tracks channels at destination
- Switch Allocation
  - For incoming virtual channel, negotiate switch on outgoing pin

Reducing routing delay: Express Cubes

- Problem: Low-dimensional networks have high k
  - Consequence: may have to travel many hops in single dimension
  - Routing latency can dominate long-distance traffic patterns
- Solution: Provide one or more "express" links



  - Like express trains, express elevators, etc
    - » Delay linear with distance, lower constant
    - » Closer to "speed of light" in medium
    - » Lower power, since no router cost
  - "Express Cubes: Improving performance of k-ary n-cube interconnection networks," Bill Dally 1991
- Another Idea: route with pass transistors through links

Summary

- 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
- Routing Algorithms restrict 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