CS184b: Computer Architecture (Abstractions and Optimizations)

Day 4: April 4, 2005
Interconnect

Previously

- CS184a
  - interconnect needs and requirements
  - basic topology
  - Mostly thought about static/offline routing

This Quarter

- This quarter
  - parallel systems require
  - typically dynamic switching
  - interfacing issues
    - model, hardware, software

Today

- Issues
- Topology/locality/scaling
  - (some review)
- Styles
  - from static
  - to online, packet, wormhole
- Online routing

Issues

Old
- Bandwidth
  - aggregate, per endpoint
  - local contention and hotspots
- Latency
- Cost (scaling)
  - locality

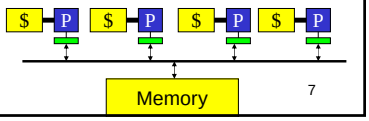
New
- Arbitration
  - conflict resolution
  - deadlock
- Routing
  - (quality vs. complexity)
- Ordering (of messages)

Topology and Locality

(Partially) Review

Simple Topologies: Bus

- Single Bus
  - simple, cheap
  - low bandwidth
    - not scale with PEs
  - typically online arbitration
    - can be offline scheduled



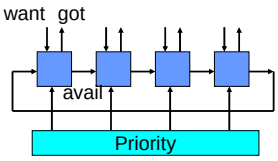
Bus Routing

- Offline:
  - divide time into N slots
  - assign positions to various communications
  - run modulo N w/ each consumer/producer send/receiving on time slot
- *e.g.*
  1: A->B
  2: C->D
  3: A->C
  4: A->B
  5: C->B
  6: D->A
  7: D->B
  8: A->D

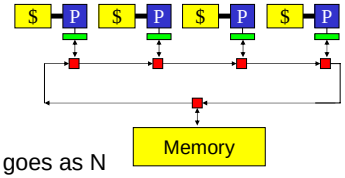
Bus Routing

- Online:
  - request bus
  - wait for acknowledge
- Priority based:
  - give to highest priority which requests
  - consider ordering
  - $Got_i = Want_i \wedge Avail_i$
    $Avail_{i+1} = Avail_i \wedge /Want_i$
- Solve arbitration in log time using parallel prefix
- For fairness
  - start priority at different node
  - use cyclic parallel prefix
    - deal with variable starting point

Arbitration Logic



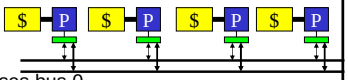
Token Ring



- On bus
  - delay of cycle goes as N
  - can't avoid, even if talking to nearest neighbor
- Token ring
  - pipeline bus data transit (ring)
    - high frequency
  - can exit early if local
  - use token to arbitrate use of bus

Multiple Busses

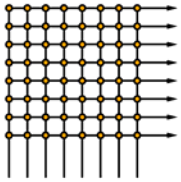
- Simple way to increase bandwidth
  - use more than one bus
- Can be static or dynamic assignment to busses



  - static
    - A->B always uses bus 0
    - C->D always uses bus 1
  - dynamic
    - arbitrate for a bus, like instruction dispatch to k identical CPU resources

Crossbar

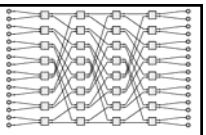
- No bandwidth reduction
  - (except receiver at endoint)
- Easy routing (on or offline)
- Scales poorly
  - N^2 area and delay
- No locality



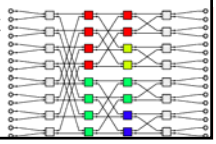
Hypercube

- Arrange $N=2^n$ nodes in n-dimensional cube
- At most n hops from source to sink
  - $N = \log_2(N)$
- High bisection bandwidth
  - good for traffic (but can you use it?)
  - bad for cost [$O(n^2)$]
- Exploit locality
- Node size grows
  - as log(N) [IO]
  - Maybe $\log^2(N)$ [xbar between dimensions]

Multistage

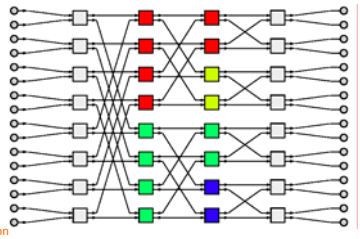


- Unroll hypercube vertices so log(N), constant size switches per hypercube node
  - solve node growth problem
  - lose locality
  - similar good/bad points for rest



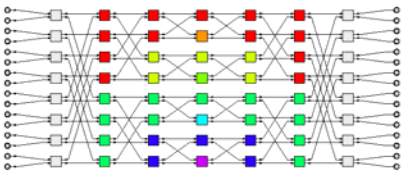
Hypercube/Multistage Blocking

- Minimum length multistage
  - many patterns cause bottlenecks
  - *e.g.*



Beneš Network

- $2\log_2(N)-1$ stages (switches in path)
- Made of N/2 2×2 switchpoints [4 sw]
- $4N \times \log_2(N)$ total switches
- Compute route in O(N log(N)) time
- Routes all permutations



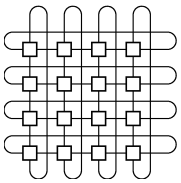
Online Hypercube Blocking

- If routing offline, can calculate Benes-like route
- Online, don't have time, or global view
- **Observation**: only a few, canonically bad patterns
- **Solution**: Route to random intermediate
  - then route from there to destination
  - …turns worst-case into average case
    - at the expense of locality

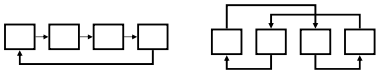
K-ary N-cube

- Alternate reduction from hypercube
  - restrict to N<log(Nodes) dimensional structure
  - allow more than 2 ordinates in each dimension
- *E.g.* mesh (2-cube), 3D-mesh (3-cube)
- Matches with physical world structure
- Bounds degree at node
- Has Locality
- Even more bottleneck potentials
  - make channels wider (CS184a:Day 17)

Torus



- Wrap around n-cube ends
  - 2-cube → cylinder
  - 3-cube → donut
- Cuts worst-case distances in half
- Can be laid-out reasonable efficiently
  - maybe 2x cost in channel width?



Fat-Tree

- Saw that communications typically has locality (CS184a)
- Modeled recursive bisection/Rent's Rule
- Leiserson showed Fat-Tree was (area, volume) universal
  - w/in log(N) the area of **any** other structure
  - exploit physical space limitations wiring in {2,3}-dimensions

MoT/Express Cube (Mesh with Bypass)

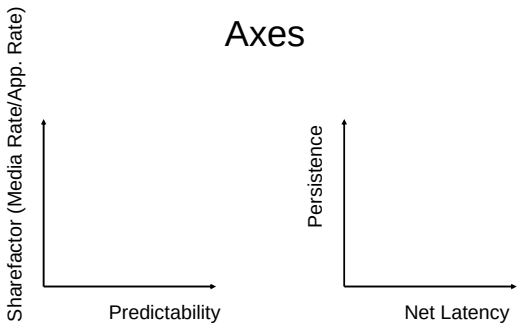
- Large machine in 2 or 3 D mesh
  - routes must go through square/cube root switches
  - vs. log(N) in fat-tree, hypercube, MIN
- Saw practically can go further than one hop on wire…
- Add long-wire bypass paths

Routing Styles

Issues/Axes

- Throughput of Communication relative to data rate of media
  - Single point-to-point link consume media BW?
  - Can share links between multiple comm streams?
  - What is the sharing factor?
- Binding time/Predictability of Interconnect
  - Pre-fab
  - Before communication then use for long time
  - Cycle-by-cycle
- Network latency vs. persistence of communication
  - Comm link persistence

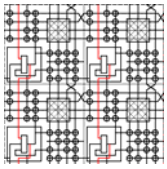
Axes



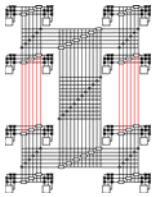
Hardwired

- Direct, fixed wire between two points
- *E.g.* Conventional gate-array, std. cell
- Efficient when:
  - know communication *a priori*
    - fixed or limited function systems
    - high load of fixed communication
      - often control in general-purpose systems
  - links carry high throughput traffic continually between fixed points

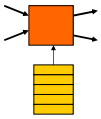
Configurable



- Offline, lock down persistent route.
- *E.g.* FPGAs
- Efficient when:
  - link carries high throughput traffic
    - (loaded usefully near capacity)
  - traffic patterns change
    - on timescale >> data transmission

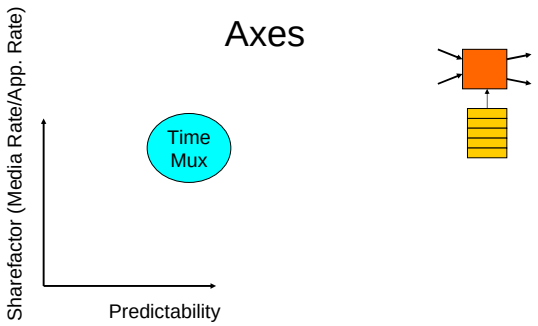


Time-Switched



- Statically scheduled, wire/switch sharing
- *E.g.* TDMA, NuMesh, TSFPGA
- Efficient when:
  - thruput per channel < thruput capacity of wires and switches
  - traffic patterns change
    - on timescale >> data transmission

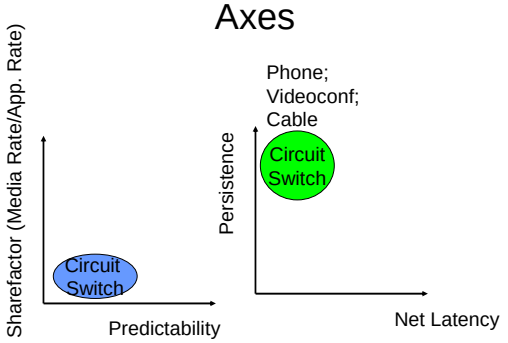
Axes



Self-Route, Circuit-Switched

- Dynamic arbitration/allocation, lock down routes
- *E.g.* METRO/RN1
- Efficient when:
  - instantaneous communication bandwidth is high (consume channel)
  - lifetime of comm. > delay through network
  - communication pattern unpredictable
  - rapid connection setup important

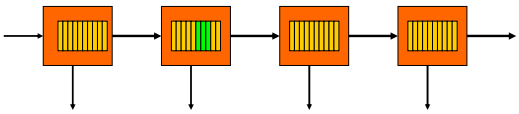
Axes



Self-Route, Store-and-Forward, Packet Switched

- Dynamic arbitration, packetized data
- Get entire packet before sending to next node
- *E.g.* nCube, early Internet routers
- Efficient when:
  - lifetime of comm < delay through net
  - communication pattern unpredictable
  - can provide buffer/consumption guarantees
  - packets small

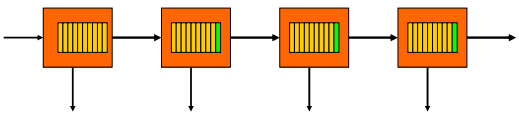
Store-and-Forward



Self-Route, Virtual Cut Through

- Dynamic arbitration, packetized data
- Start forwarding to next node as soon as have header
- Don't pay full latency of storing packet
- Keep space to buffer entire packet if necessary
- Efficient when:
  - lifetime of comm < delay through net
  - communication pattern unpredictable
  - can provide buffer/consumption guarantees
  - packets small

Virtual Cut Through

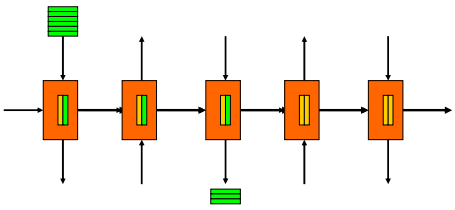


Three words from same packet

Self-Route, Wormhole Packet-Switched

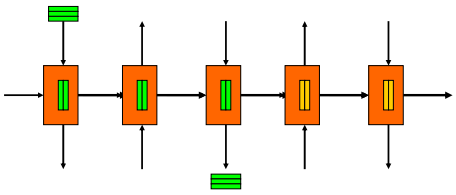
- Dynamic arbitration, packetized data
- *E.g.* Caltech MRC, Modern Internet Routers
- Efficient when:
  - lifetime of comm < delay through net
  - communication pattern unpredictable
  - can provide buffer/consumption guarantees
  - message > buffer length
    - allow variable (? Long) sized messages

Wormhole



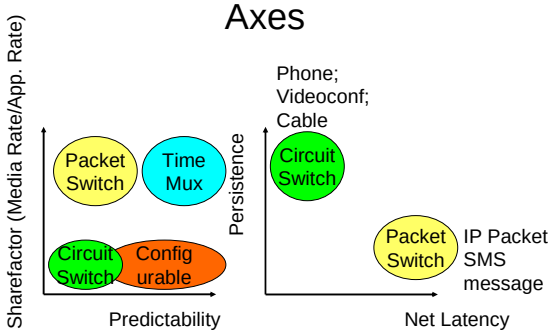
Single Packet spread through net when not stalled

Wormhole



Single Packet spread through net when stalled.

Axes



Online Routing

Costs: Area

- Area
  - switch (1-1.5Kλ²/ switch)
    - larger with pipeline (4Kλ²) and rebuffer
  - state (SRAM bit = 1.2Kλ²/ bit)
    - multiple in time-switched cases
  - arbitrartion/decision making
    - usually dominates above
  - buffering (SRAM cell per buffer)
    - can dominate

Time Mux

Dynamic

Area

$$A_{switching} = W \times Ins \times Outs \times A_{sw}$$

$$A_{inst} = Outs \times \log_2(Ins) \times Cycles \times A_{bit}$$

$$A_{arb} = Ins \times Outs \times A_{arbitration}$$

$$A_{queue} = W \times (Ins + Outs) \times Depth \times A_{bit}$$

(queue rough approx; you will refine)

Costs: Latency

- Time single path
  - make decisions
  - round-trip flow-control
- Time contention/traffic
  - blocking in buffers
  - quality of decision
    - pick wrong path
    - have stale data

Intermediate Approach

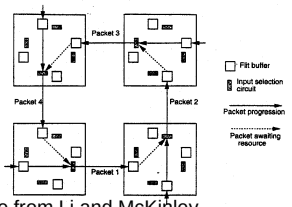
- For large # of predictable patterns
  - switching memory may dominate allocation area
  - area of routed case < time-switched
  - [*e.g.* Large Cycles]
- Get offline, global planning advantage
  - by **source routing**
    - source specifies offline determined route path
    - offline plan avoids contention

Offline vs. Online

- If know patterns in advance
  - offline cheaper
    - no arbitration (area, time)
    - no buffering
    - use more global data
      - better results
- As becomes less predictable
  - benefit to online routing

Deadlock

- Possible to introduce deadlock
- Consider wormhole routed mesh



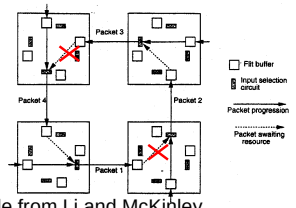
[example from Li and McKinley, IEEE Computer v26n2, 1993]

Dimension Order Routing

- Simple (early Caltech) solution
  - order dimensions
  - force complete routing in lower dimensions before route in next higher dimension

Dimension Ordered Routing

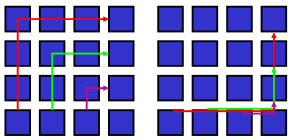
- Route Y, then Route X



[example from Li and McKinley, IEEE Computer v26n2, 1993]

Dimension Order Routing

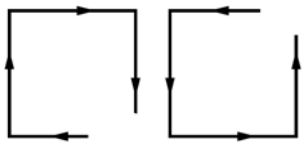
- Avoids cycles in channel graph
- Limits routing freedom
- Can cause artificial congestion
  - consider
    - (0,0) to (3,3)
    - (1,0) to (3,2)
    - (2,0) to (3,1)



- [There is a rich literature on how to do better]

Turn Model

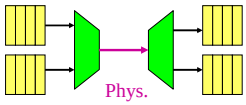
- Problem is cycles
- Selectively disallow turns to break cycles
- 2D Mesh



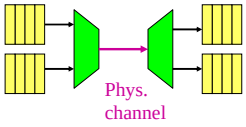
West-First Routing

Virtual Channel

- **Variation**: each physical channel represents multiple logical channels
  - each logical channel has own buffers
  - blocking in one VC allows other VCs to use the physical link



Virtual Channels



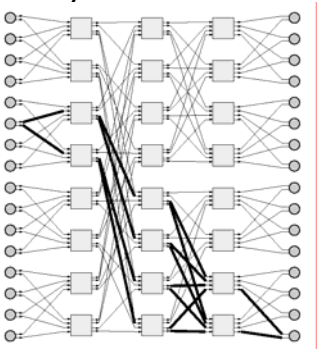
- Benefits
  - can be used to remove cycles
    - *e.g.* separate increasing and decreasing channels
    - route increasing first, then decreasing
    - more freedom than dimension ordered
  - prioritize traffic
    - *e.g.* prevent control/OS traffic from being blocked by user traffic
  - better utilization of physical routing channels

Lost Freedom?

- Online routes often make (must make) decisions based on local information
- Can make wrong decision
  - *i.e.* two paths look equally good at one point in net
    - but one leads to congestion/blocking further ahead

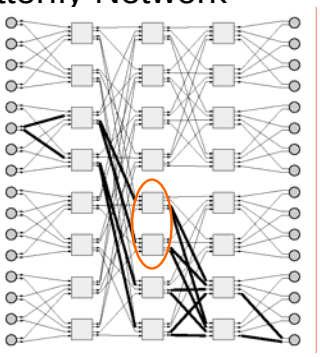
Multibutterfly Network

- Dilated routers
  - have multiple outputs in each logical direction
  - Means multiple paths between any src,sink pair
- Use to avoid congestion
  - also faults



Multibutterfly Network

- Can get into local blocking when there is a path
- Costs of not having global information



Transit/Metro

- Self-routing circuit switched network
- When have choice
  - select randomly
    - avoid bad structural cases
- When blocked
  - drop connection
  - allow to route again from source
  - stochastic search explores all paths
    - finds any available

Relation to Project

Intuitive Tradeoff

- Benefit of Time-Multiplexing?
  - Minimum end-to-end latency
  - No added decision latency at runtime
  - Offline route → high quality route
    - → use wires efficiently
- Cost of Time-Multiplexing?
  - Route task must be static
    - Cannot exploit low activity
  - Need memory bit per switch per time step
    - Lots of memory if need large number of time steps…

Intuitive Tradeoff

- Benefit of Packet Switching?
  - No area proportional to time steps
  - Route only active connections
  - Avoids slow, off-line routing
- Cost of Packet Switching?
  - Online decision making
    - Maybe won't use wires as well
    - Potentially slower routing?
      - Slower clock, more clocks across net
  - Data will be blocked in network
    - Adds latency
    - Requires packet queues

Packet Switch Motivations

- SMVM:
  - Long offline routing time limits applicability
  - Route memory exceeds compute memory for large matricies
- ConceptNet:
  - Evidence of low activity for keyword retrieval … could be important to exploit

Example

- ConceptNet retrieval
  - Visits 84K nodes across all time steps
  - 150K nodes
  - 8 steps → 1.2M node visits
  - Activity less than 7%

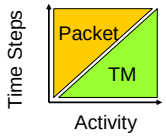
Dishoom/ConceptNet Estimates

N_z	N_{FPGA}	T_{comp}	T_{load}	T_{lat}	T_{step}
1	64	500	1500	48	2048
2	128	250	750	52	1052
4	256	125	375	60	560
8	512	63	188	76	327

Pushing all nodes, all edges;
Bandwidth (T_{load}) dominates.

Question

- For what activity factor does Packet Switching beat Time Multiplexed Routing?
  - To what extent is this also a function of total time steps?



Admin

- Reading
- VHDL intro on Wednesday
- Fast Virtex Queue Implementations on Friday

Big Ideas

- Must work with constraints of physical world
  - only have 3 dimensions (2 on current VLSI) in which to build interconnect
  - Interconnect can be dominate area, time
  - gives rise to universal networks
    - *e.g.* fat-tree

Big Ideas

- Structure
  - exploit physical locality where possible
  - the more predictable behavior
    - cheaper the solution
  - exploit earlier binding time
    - cheaper configured solutions
    - allow higher quality offline solutions
- Interconnect style
  - Driven by technology and application traffic patterns