ON-CHIP NETWORK INNOVATIONS

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing
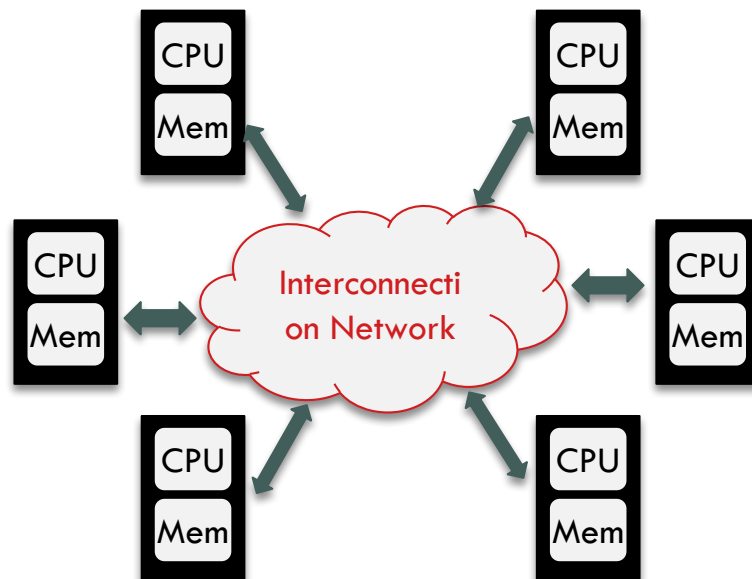
University of Utah

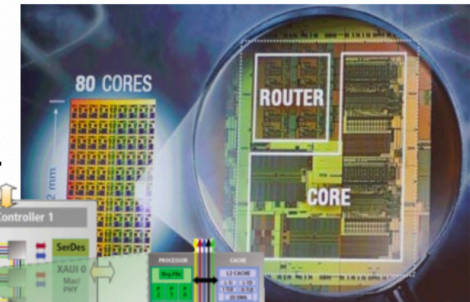# Overview

- Upcoming deadline
  - Feb.3rd: project group formation
  - No groups have sent me emails!
- This lecture
  - Basics of the interconnection networks
  - Network topologies
  - Flow control
  - Routing algorithm
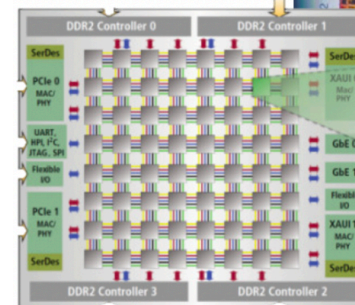 - Emerging on-chip networks

On-chip Interconnection Networks

☐ An infrastructure connecting various components in current and future ICs
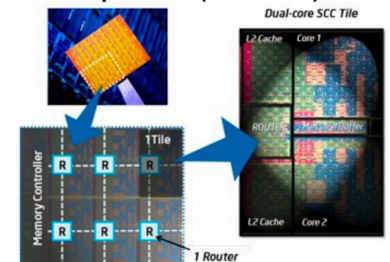

Intel 80-core (2007 ISSCC)

Tilera Tile64

Intel Single-Chip Cloud Computer (SCCC)

**Mesh is mostly employed due to its scalability.**

Network Topology

Network Topologies

- Regular vs. irregular graphs
  - Examples of regular networks are mesh and ring

- Distances in the network
  - Routing distance: number of links/hops along a route
  - Network diameter: maximum number of hops per route
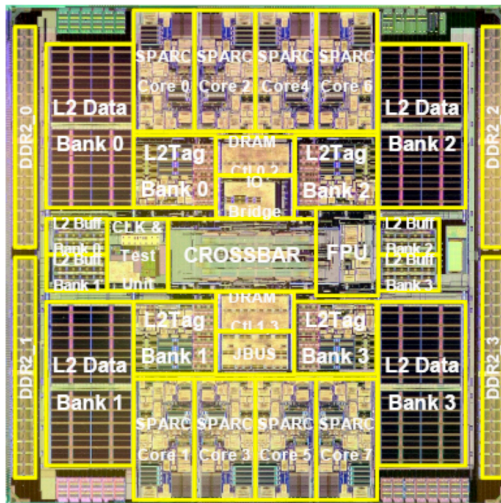 - Average distance: average number of links/hops across all valid routes

Example Topologies

- Bus
  - Simple structure; efficient for small number of nodes
  - Not scalable; highly contended
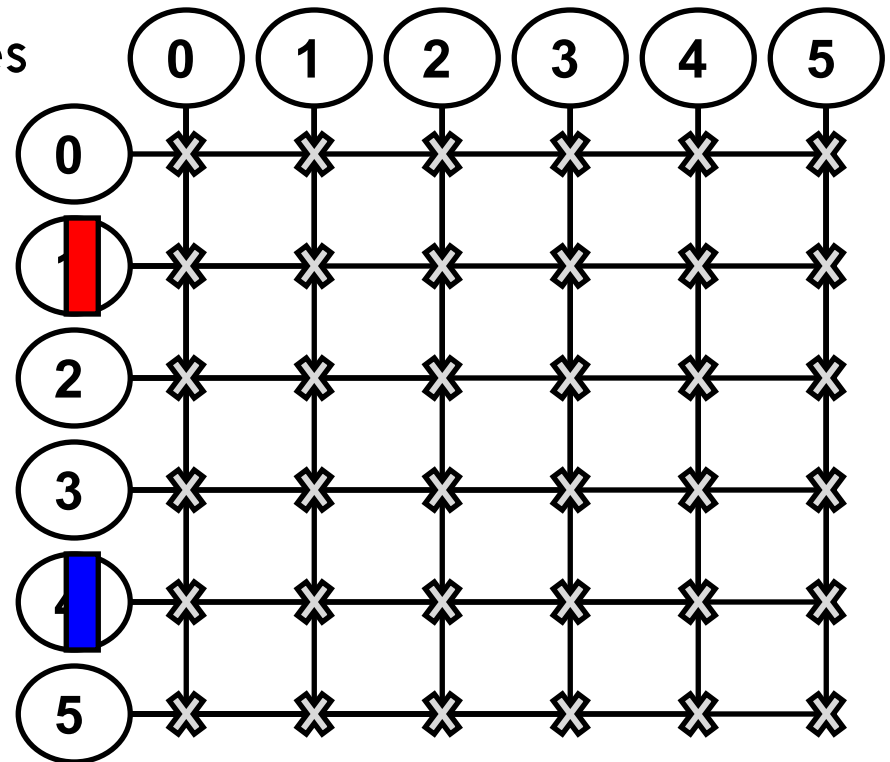 - Used in many processors

Bus

Point to Point
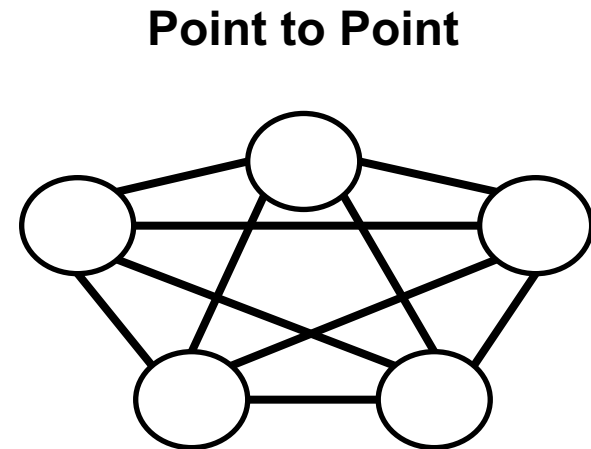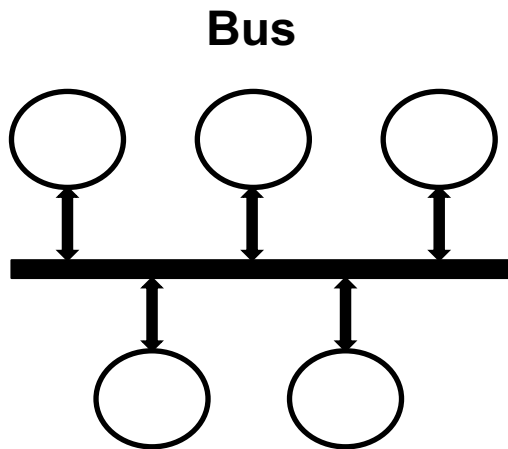
Example Topologies

- Crossbar
  - Complex arbitration
  - High throughput and fast
  - Requires a lot of resources
  - Used in Sun Niagara I/II

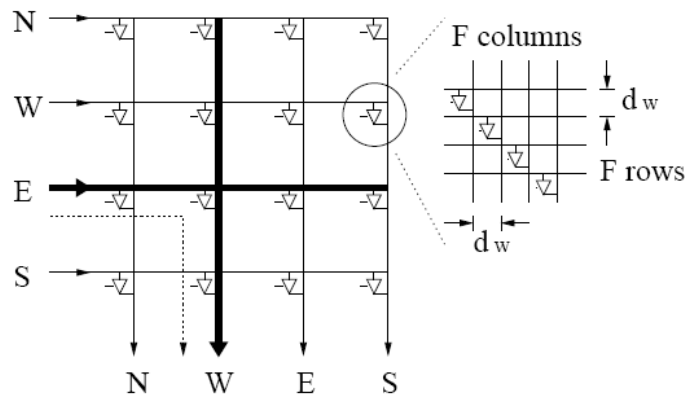
[UltraSPARC T1]
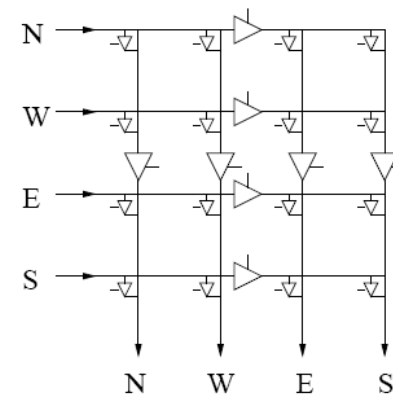
Example Topologies

□ Segmented crossbar

  ◘ Reduce switching capacitance (~15-30%)

  ◘ Need a few additional signals to control tri-states
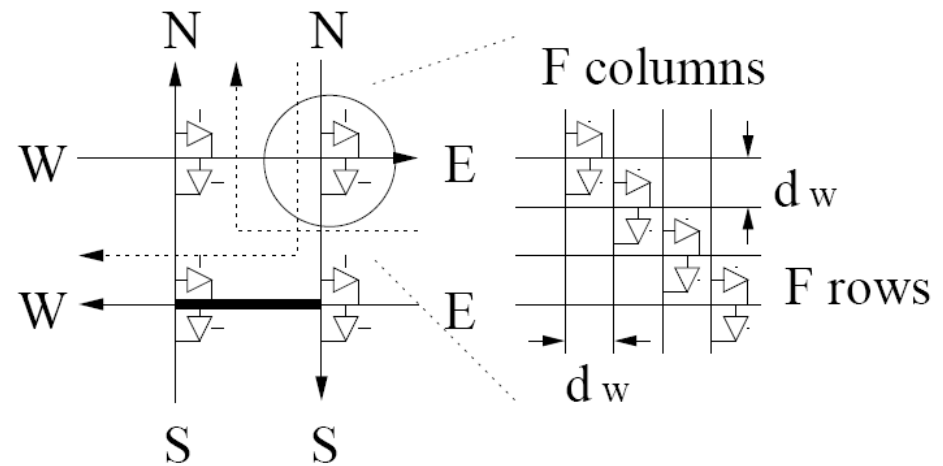

(a) A 4×4 matrix crossbar.

(b) A 4×4 segmented cross-bar with 2 segments per line.

[Wang'03]

Example Topologies

- Goal: optimize for the common case
  - Straight-through traffic does not go thru tristate buffers

- Some combinations of turns are not allowed
  - Why?

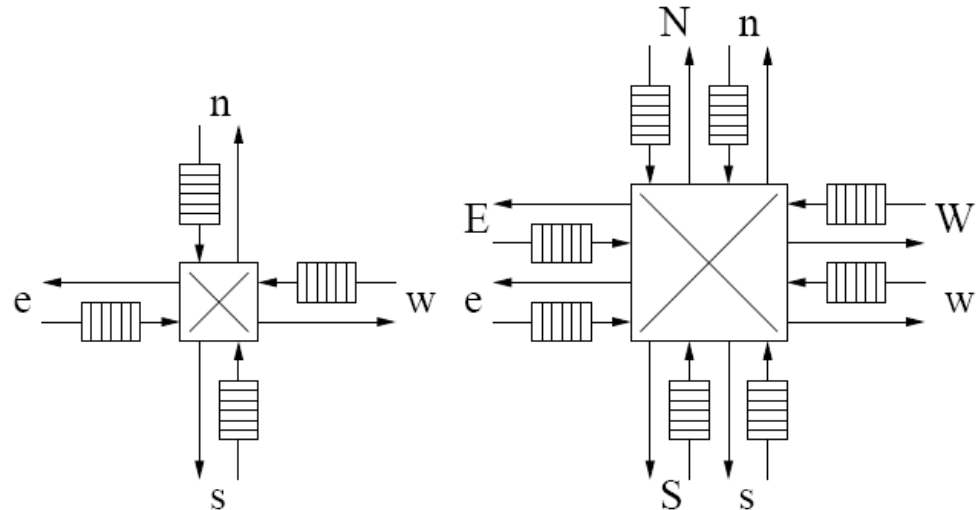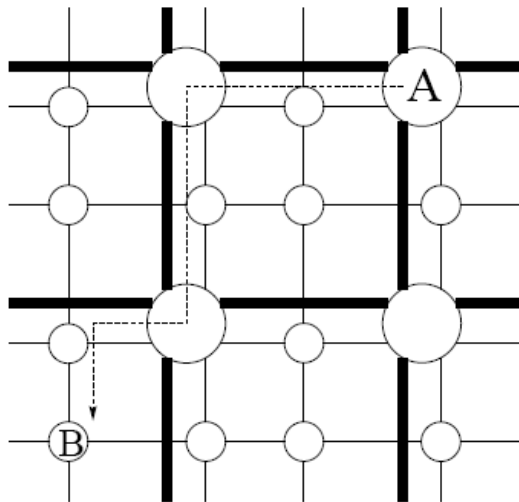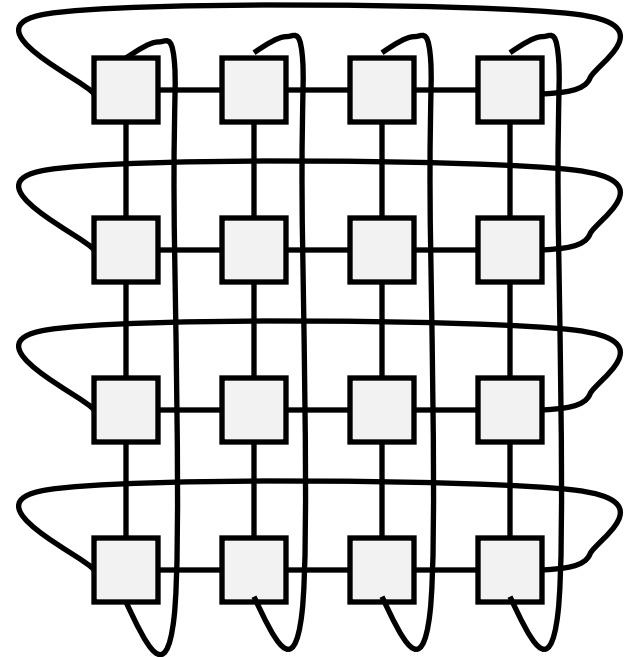**Read the paper for details.**

(a) A 4×4 cut-through crossbar.

[Wang'03]

Example Topologies

- Express channels to reduce number of hops
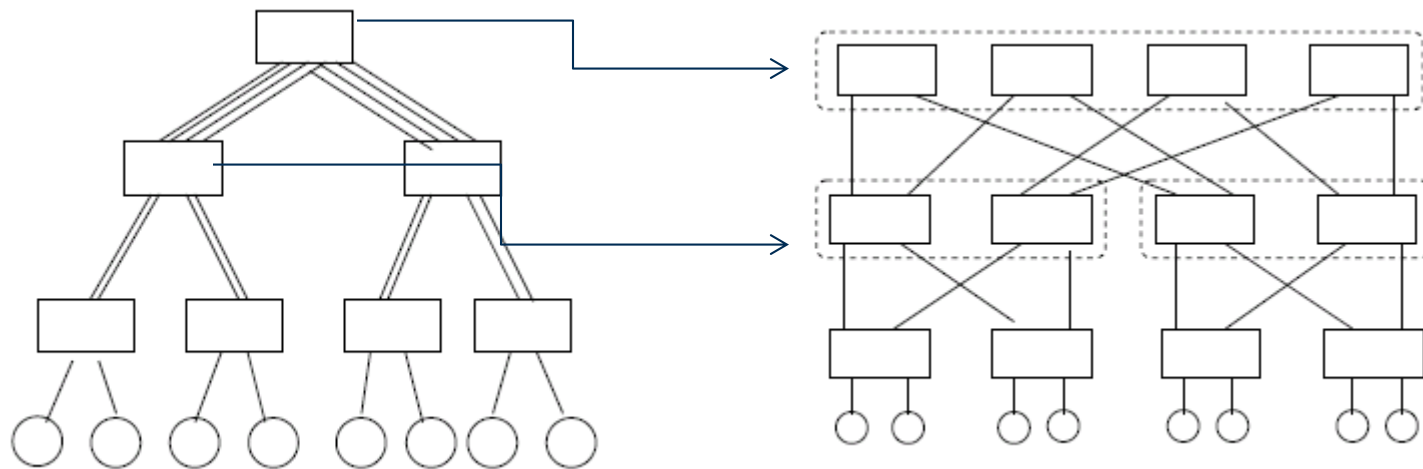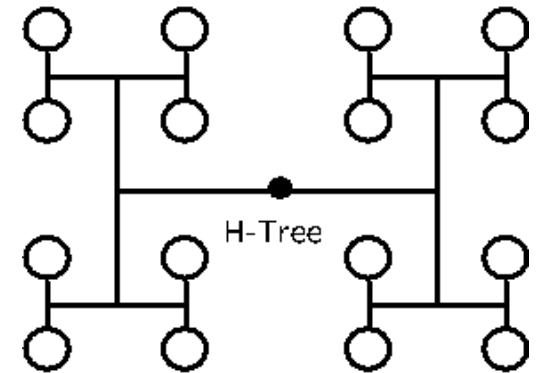 - like taking the freeway

[Wang'03]

Example Topologies

- Ring
  - Cheap; long latency
  - IBM Cell
- Mesh
  - Path diversity, efficient
  - Tilera 100-core
- Torus
  - More path diversity
  - Expensive and complex

Example Topologies

- Tree

  - Simple and low cost

  - Easy to layout

  - Efficiently handles local traffic

  - Towards root, links are heavily contended

Example Topologies

- Omega network
  - Single path from source to destination

  - Does not support all possible permutations

  - Proposed to replace costly crossbars as processor-memory interconnect
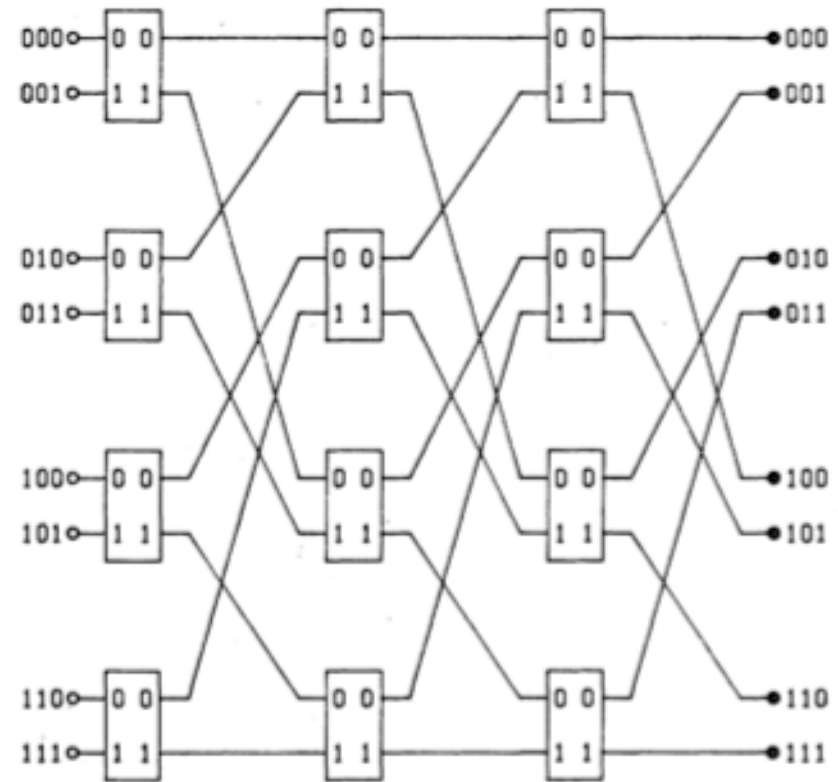
Fig. 2. Omega-network ($N = 8$).

[Gottlieb'82]

Flow Control

Sending Data in Network

- Circuit switching
  - Establish full path; then send data
  - Everyone else using the same link has to wait
  - Setup overheads

- Packet switching
  - Route individual packets (via different paths)
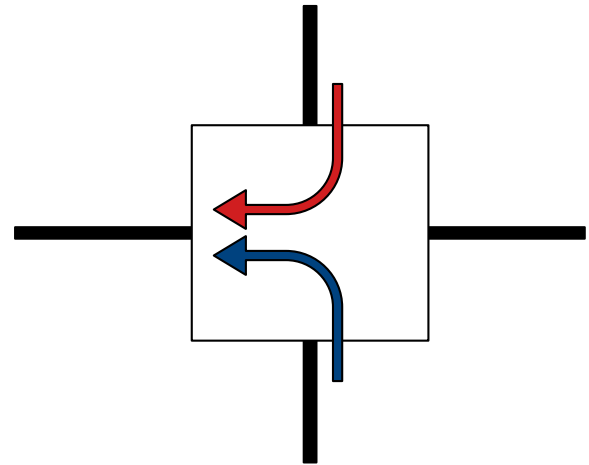 - More flexible than CS
  - May be slower than CS

Handling Contention

□ Problem
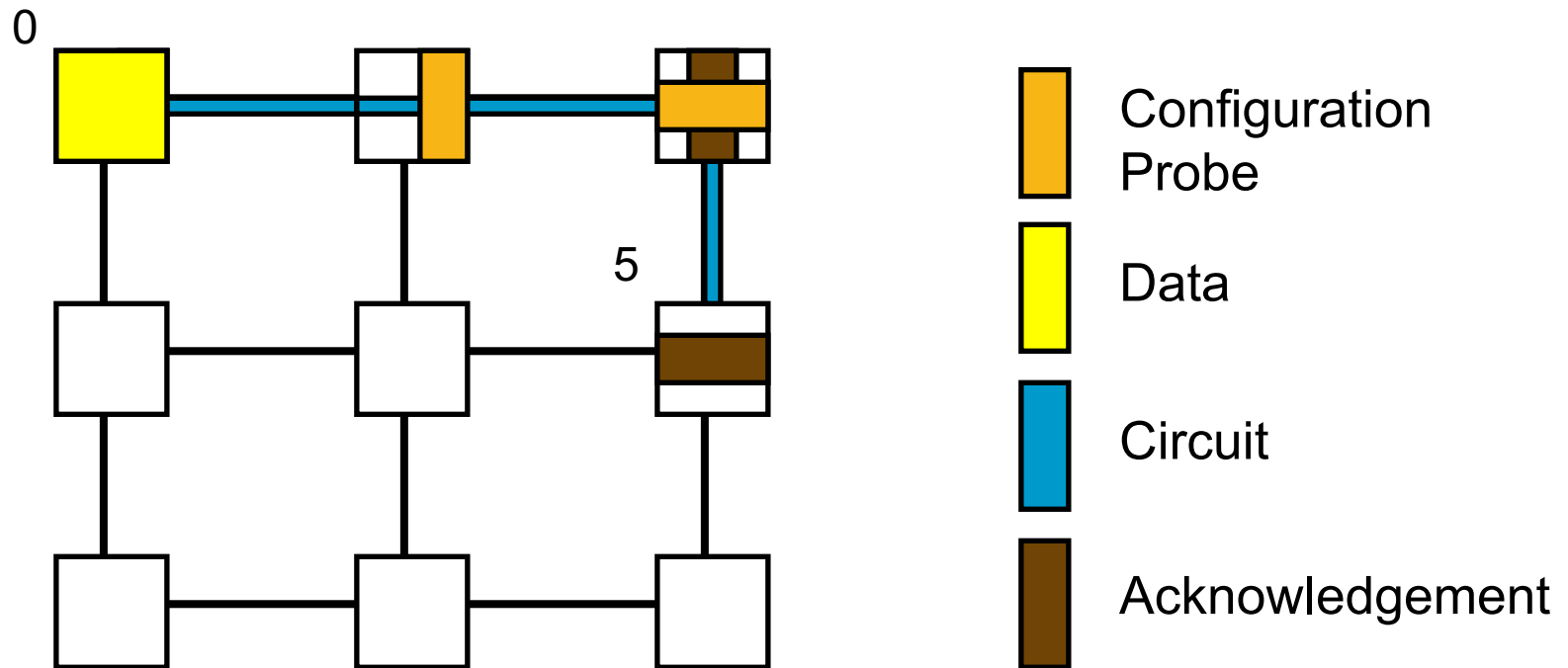
 ◻ Two packets want to use the same link at the same time

□ Possible solutions

 ◻ Drop one

 ◻ Misroute one (deflection)
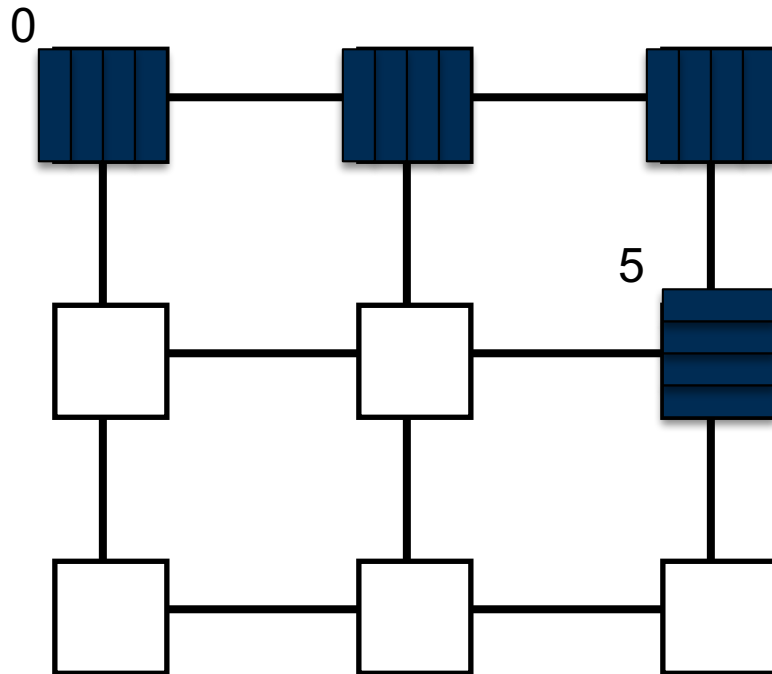
 ◻ Buffer one

Circuit Switching Example

- Significant latency overhead prior to data transfer
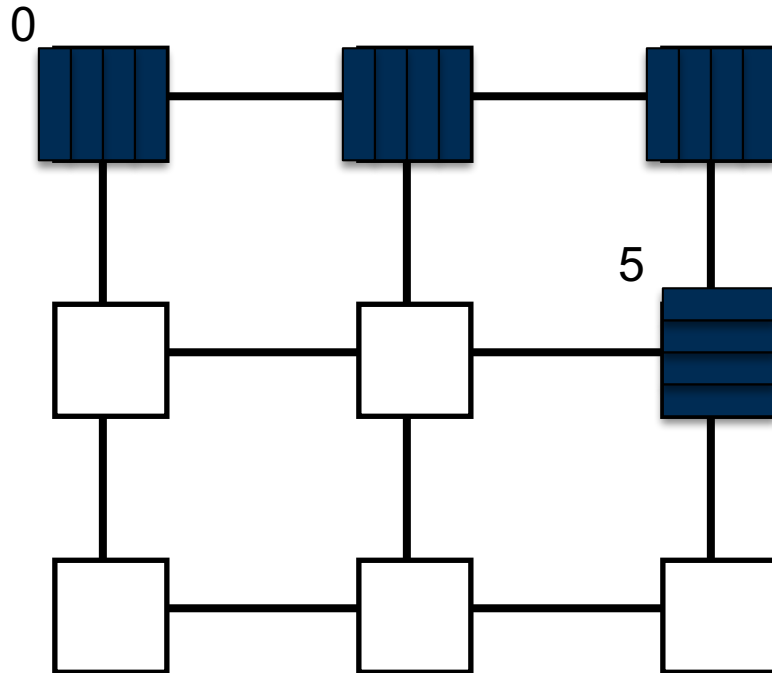- Other requests forced to wait for resources

[Lipasti]

Store and Forward Example
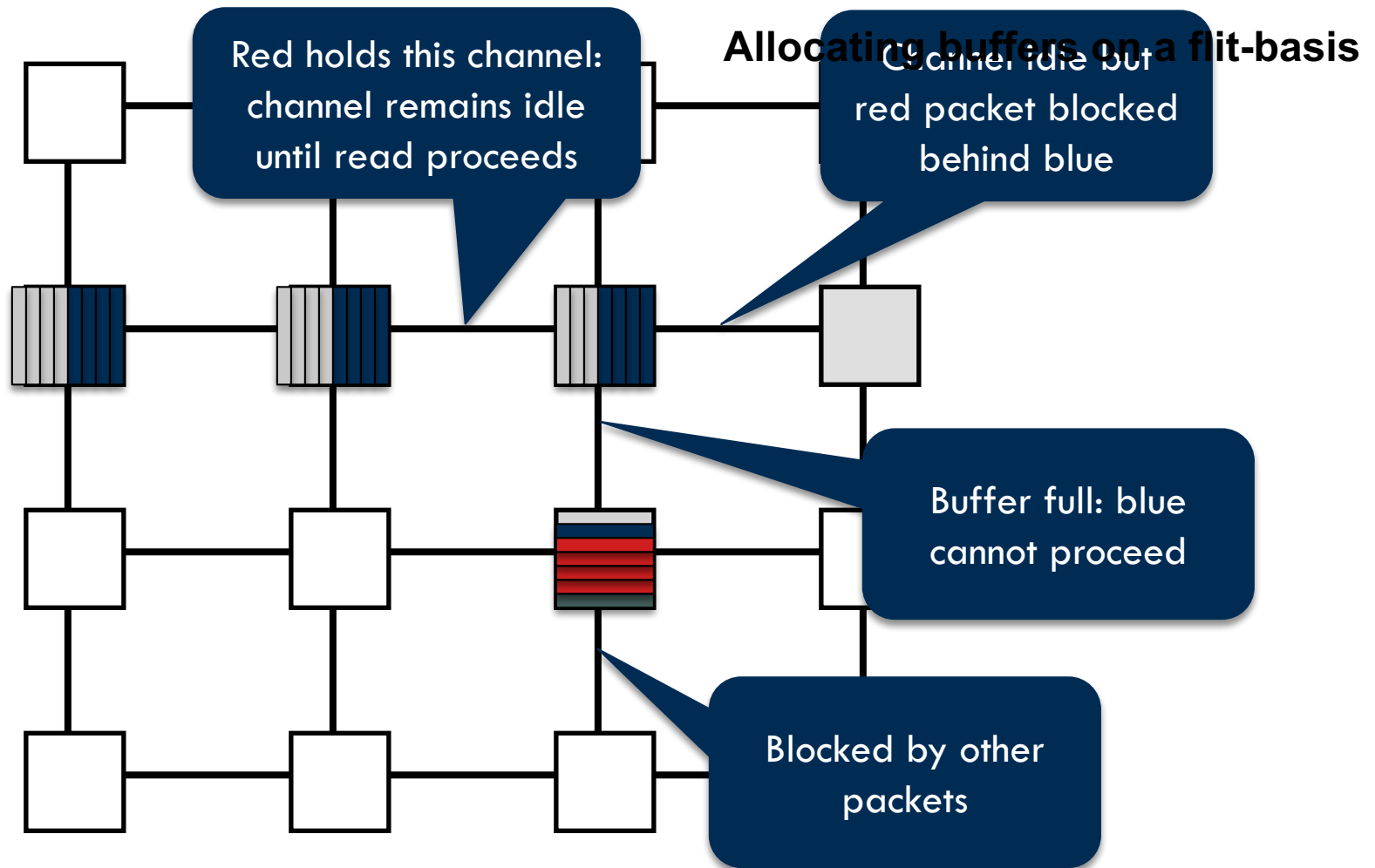
- High per-hop latency

- Larger buffering required
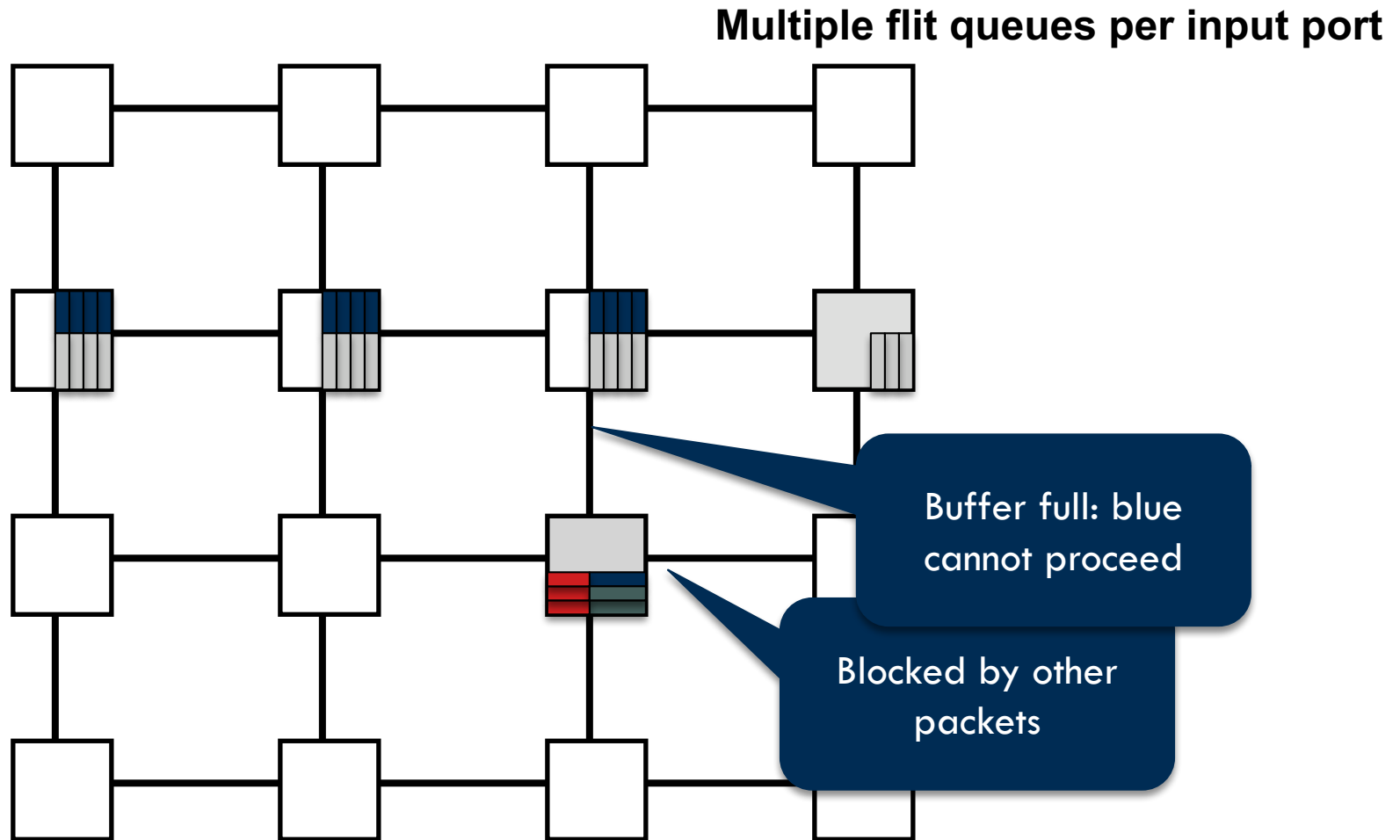
0

5

[Lipasti]

Virtual Cut Through Example

- Lower per-hop latency
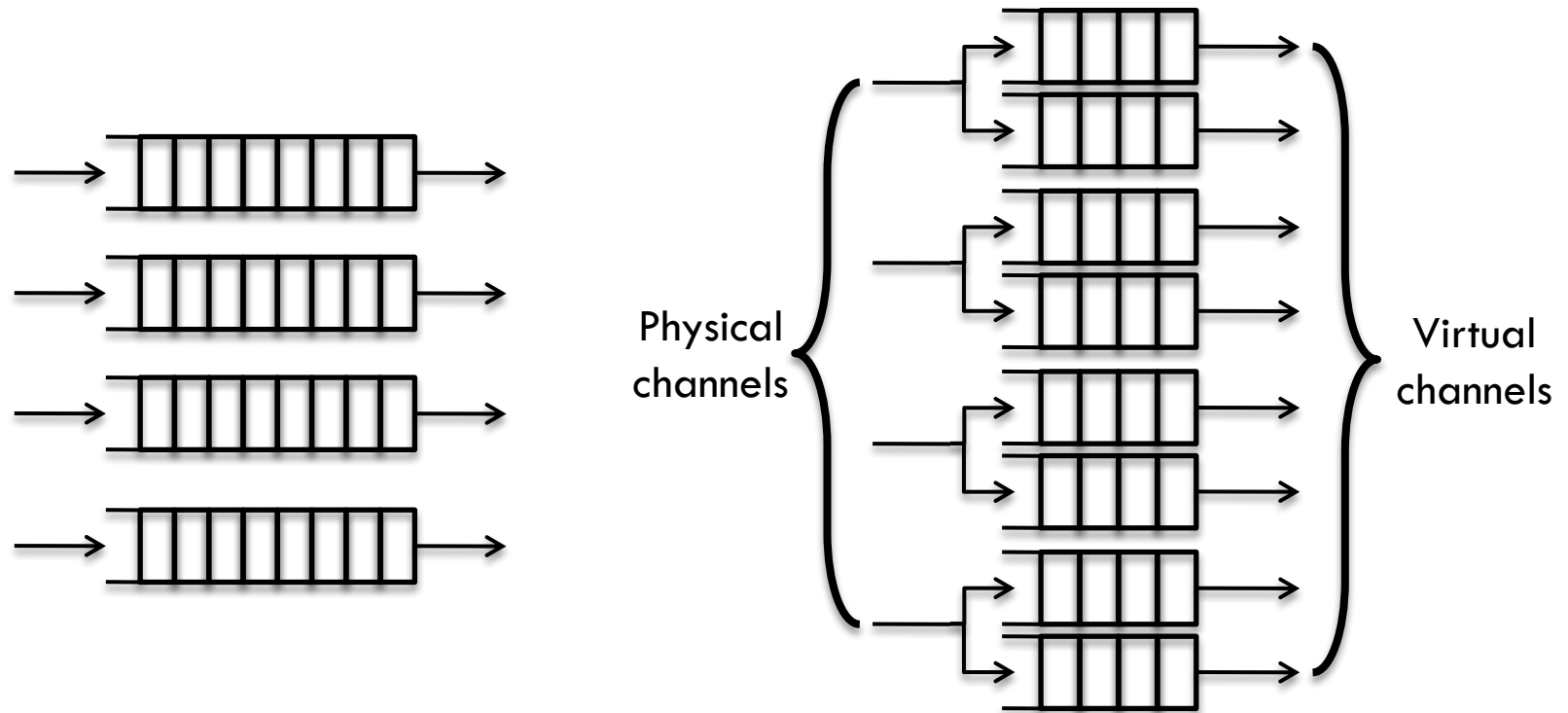- Larger buffering required

[Lipasti]

Wormhole Example

Virtual Channel Example

[Lipasti]

Virtual Channel Buffers

- Single buffer per input
- Multiple fixed length queues per physical channel

Physical channels

Virtual channels

[Lipasti]

Routing Algorithm

Types of Routing Algorithms

☐ Deterministic

  ◘ Always chooses the same path for a communicating source-destination pair

☐ Oblivious

  ◘ Chooses different paths, without considering network state

☐ Adaptive

  ◘ Can choose different paths, adapting to the state of the network

Deterministic Routing

- All packets between the same (source, destination) pair take the same path

- Dimension-order routing
  - E.g., XY routing (used in Cray T3D, and many on-chip networks)

- First traverse dimension X, then traverse dimension Y

- Deadlock freedom

- Could lead to high contention

Oblivious Routing

□ Valiant's Algorithm
  ▫ randomly choose intermediate node d'
  ▫ Route from s to d' and from d' to d.
□ Randomizes any traffic pattern
  ▫ Balances network load
  ▫ Non-minimal
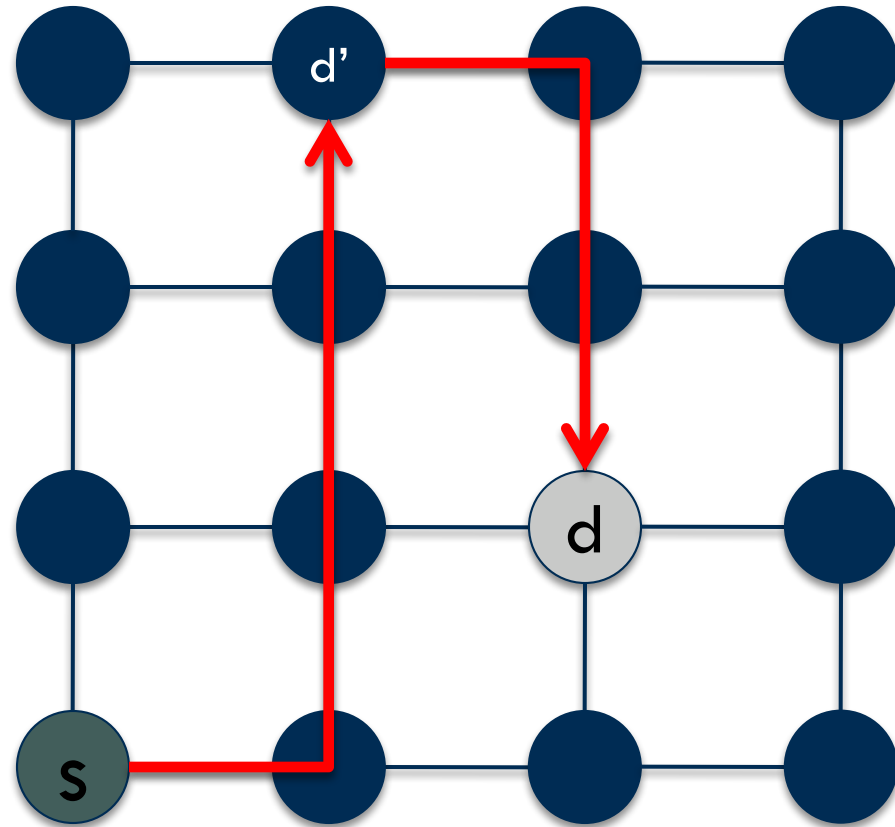
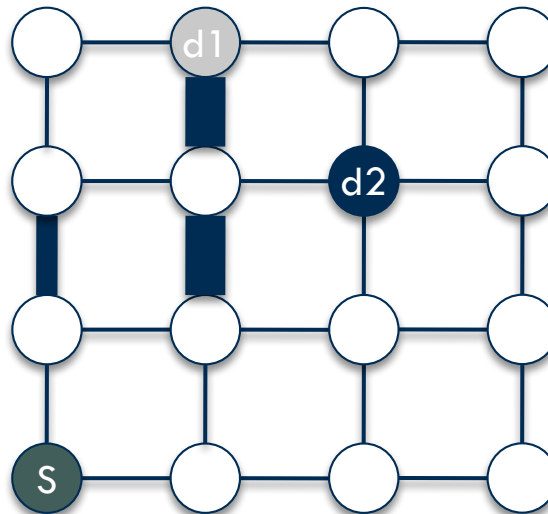# Oblivious Routing

- Minimal Oblivious
  - d' must lie within minimum quadrant
  - 6 options for d'
  - Only 3 different paths
- Achieve some load balancing, but use shortest paths
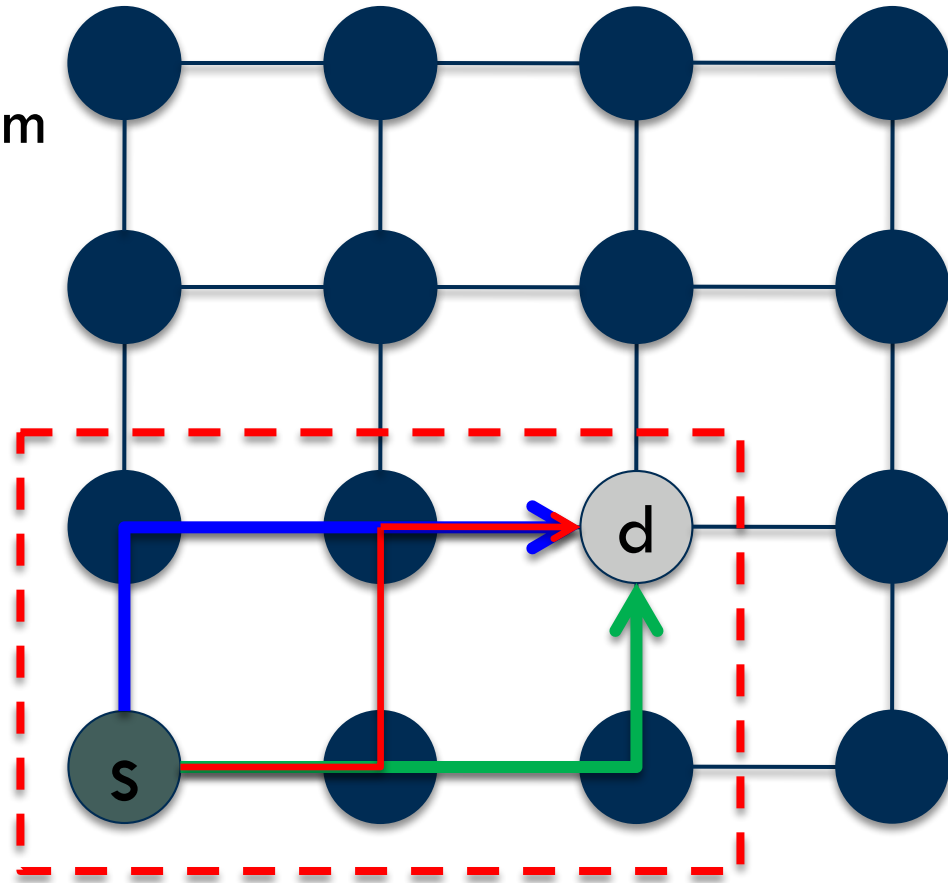
Adaptive Routing

- Make decisions according to the current state of the network

- Local vs. global information
  - Local states are available easily
 - Global information more expensive

Deadlock

- No forward progress

- Caused by circular dependencies on resources

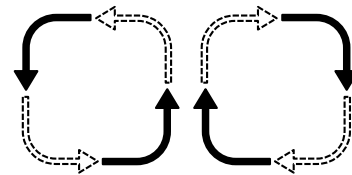- Each packet waits for a buffer occupied by another packet downstream
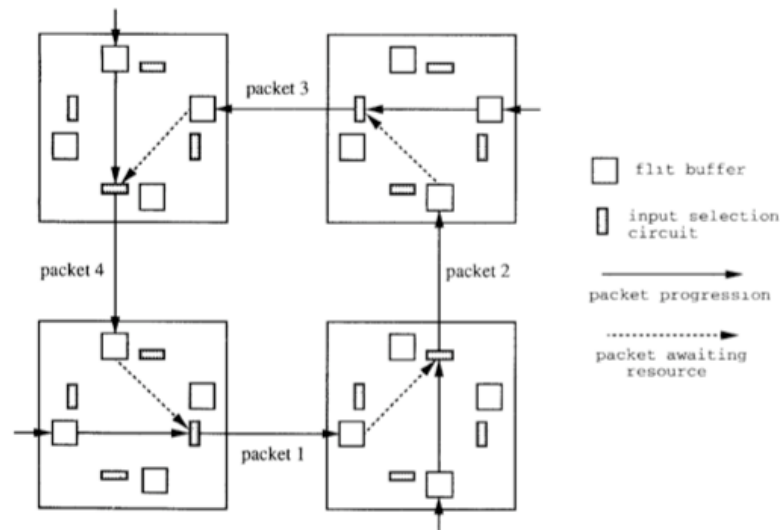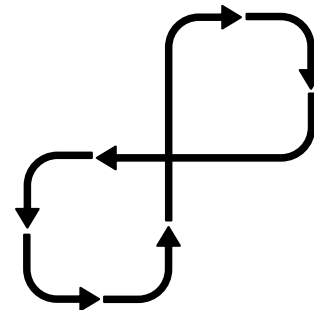

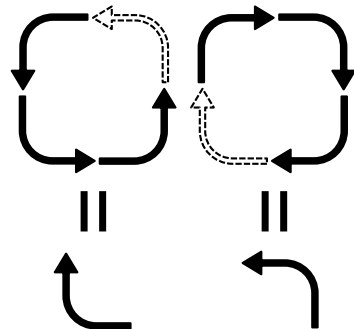*[Glass'92]*

Handling Deadlock

- Analyze directions in which packets can turn in the network

- Determine the cycles that such turns can form

- Prohibit just enough turns to break possible cycles
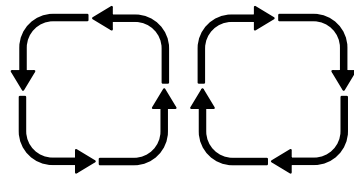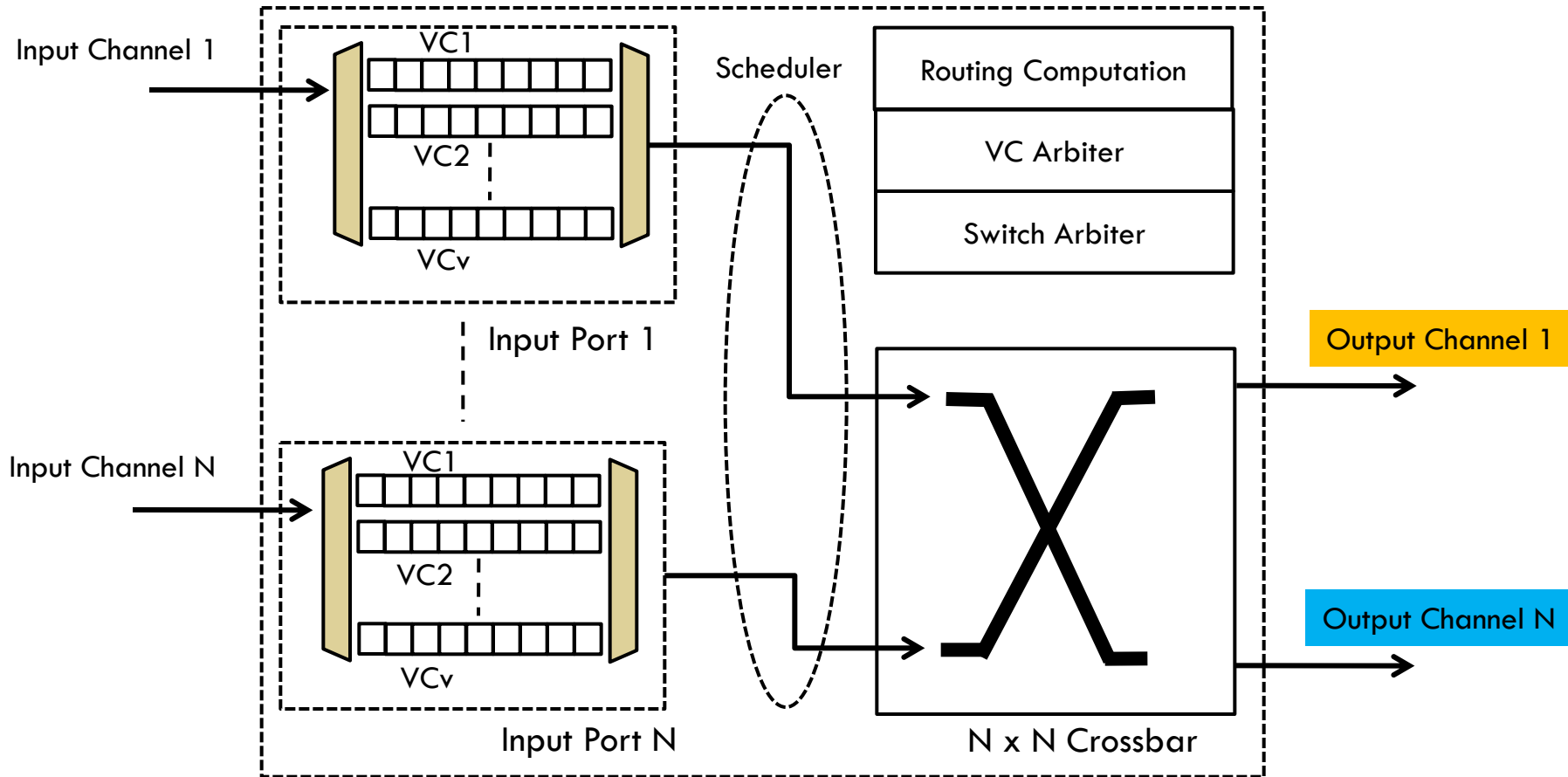
Cycles in 2D mesh   **The 4 allowed turns**
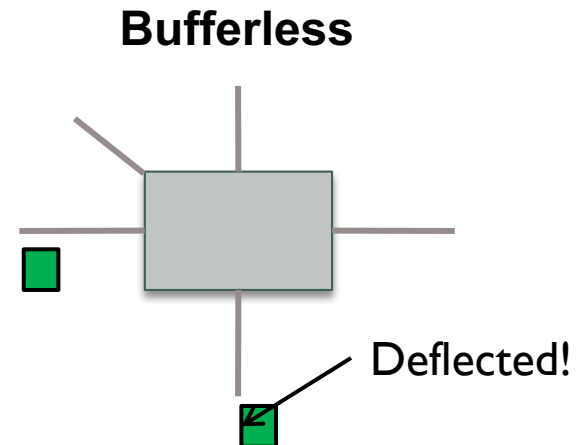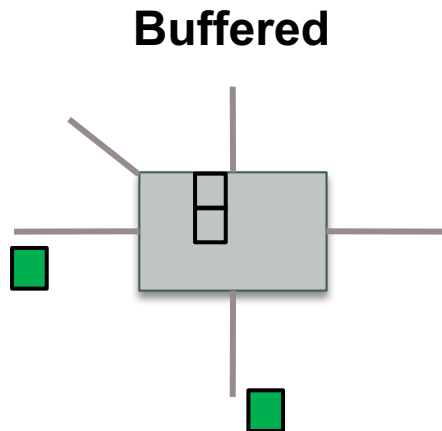
[Glass'92]

A Typical Router Architecture

Buffer-less Routing

□ Routing buffers

  ❑ necessary for high throughput routing

  ❑ consume significant chip area and power
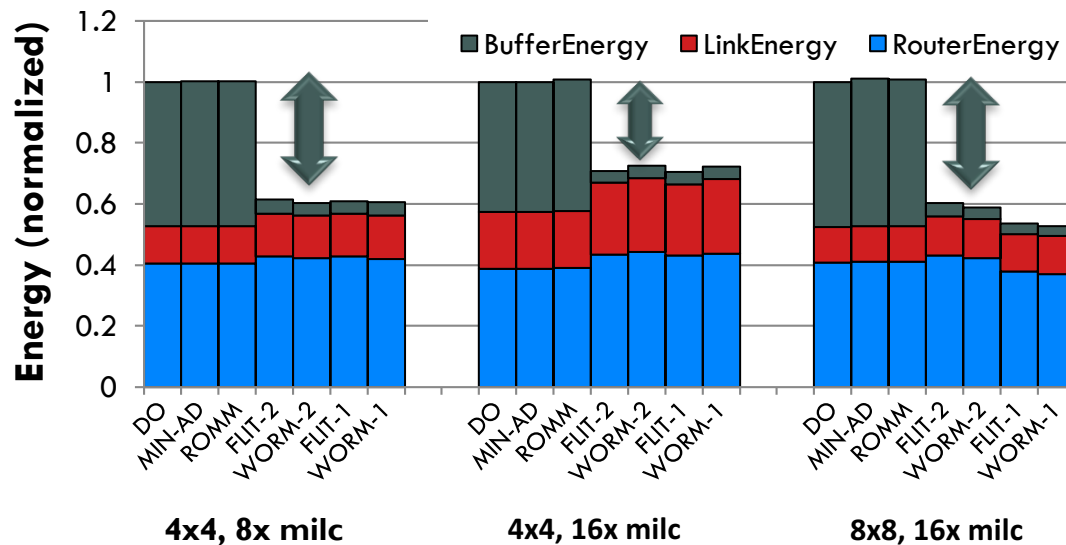
    ◾ 75% of die area in TRIPS IC [Gratz'06]

Problem: packets may be deflected forever (livelock)

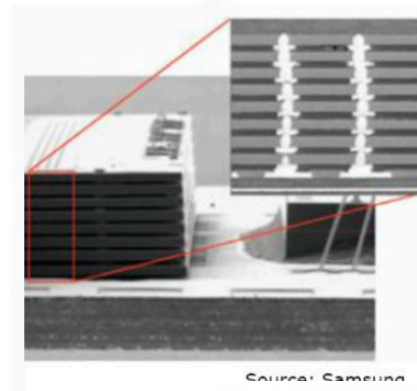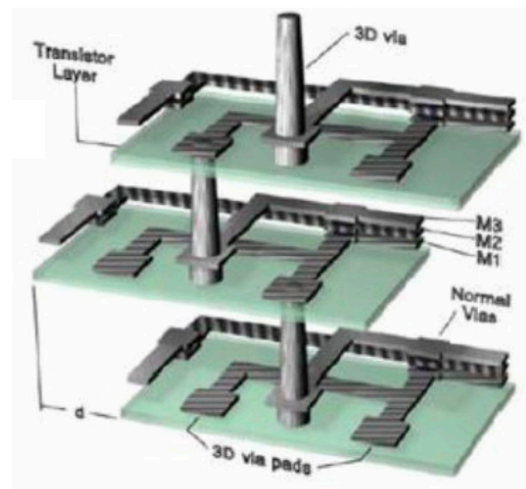**Buffered**   **Bufferless**

Deflected!

[Moscibroda'09]

Buffer-less Routing

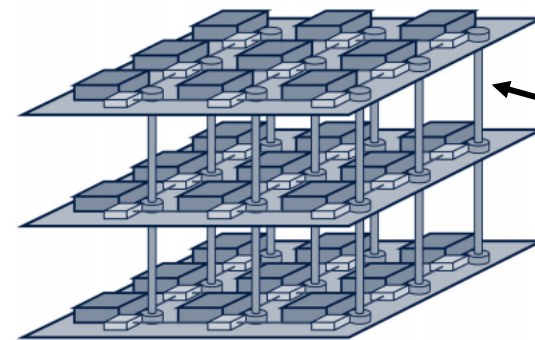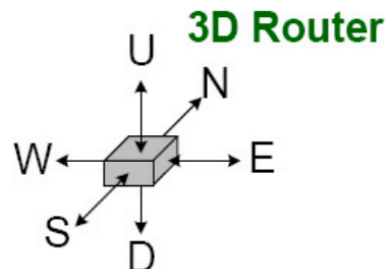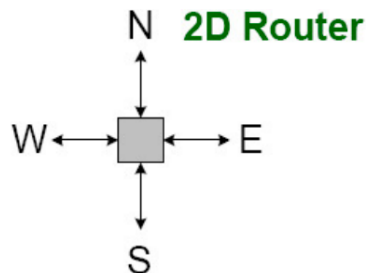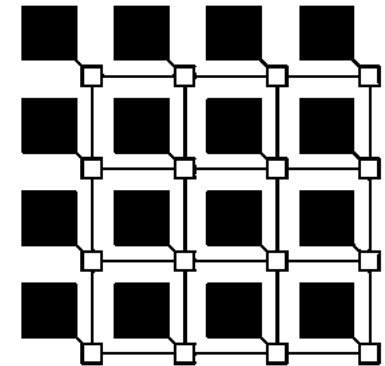☐ Significant energy improvements (almost 40%)

[Moscibroda'09]

Networks for 3D Architectures

3D NOC Architectures

- Interconnection networks using die-stacking technology

2D Mesh Network

2D Router

3D Router

Through Silicon Via (TSV)

Stacked layers   *[Feero'09]*

Thermal Challenges

- Power consumption is more challenging in 3D chips
  - Longer heat dissipation paths
  - More transistors on chip; larger power density

- Resultant issues for 3D ICs
  - Higher temperature; more leakage
  - New set of reliability issues
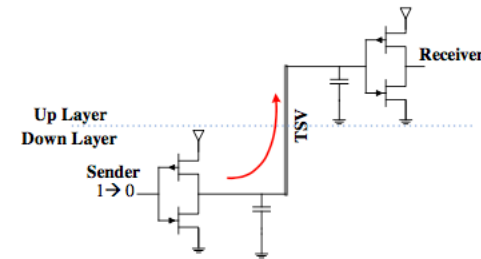 - Performance degradation

Current Flow in TSVs

- Current flow is data dependent
- Every voltage level switching in a TSV consumes energy
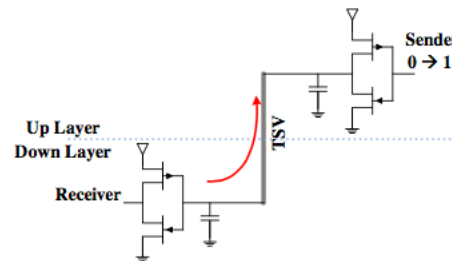- TSV switching has inductive effects
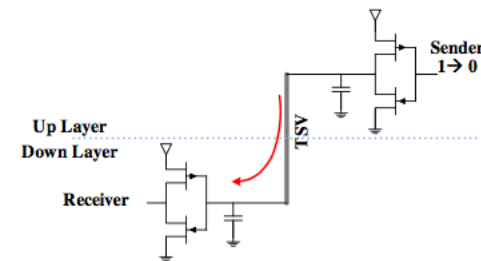
Can we reduce switching activity of TSVs?
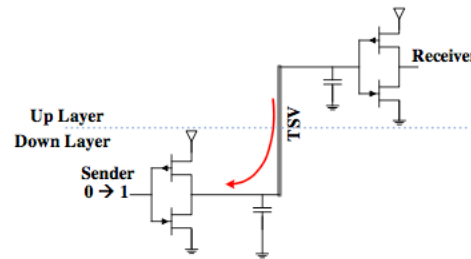
(a) Downward current flow

(b) Upward current flow

(c) Upward current flow

(d) Downward current flow
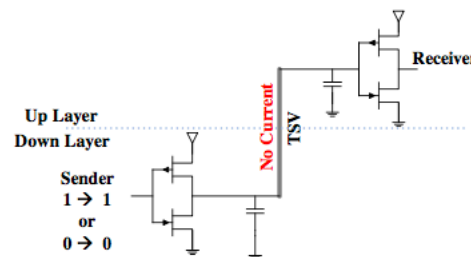
(e) Off-Current mode

(f) Off-Current mode

[Eghbal'14]
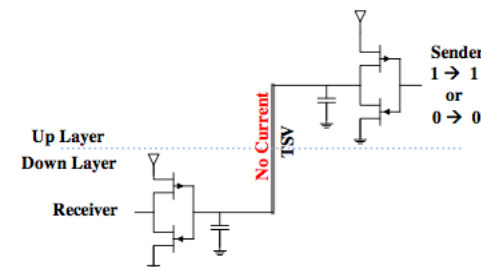
Multi-layer Router Architecture

- Observation: many of the data flits (up to 60% of CMP Cache Data from real workloads) have frequent patterns such as all zeros or all ones

- Split router comps (crossbar, buffer, etc.) in the third dimension, and the consequent vertical interconnect (via) design overheads.

(a) 2DB, 3DB

(b) 3DM (full)

[Park'08]

Summary of Possible Optimizations

□ Architectural solutions for thermal issues

◻ Thermal-aware application layout

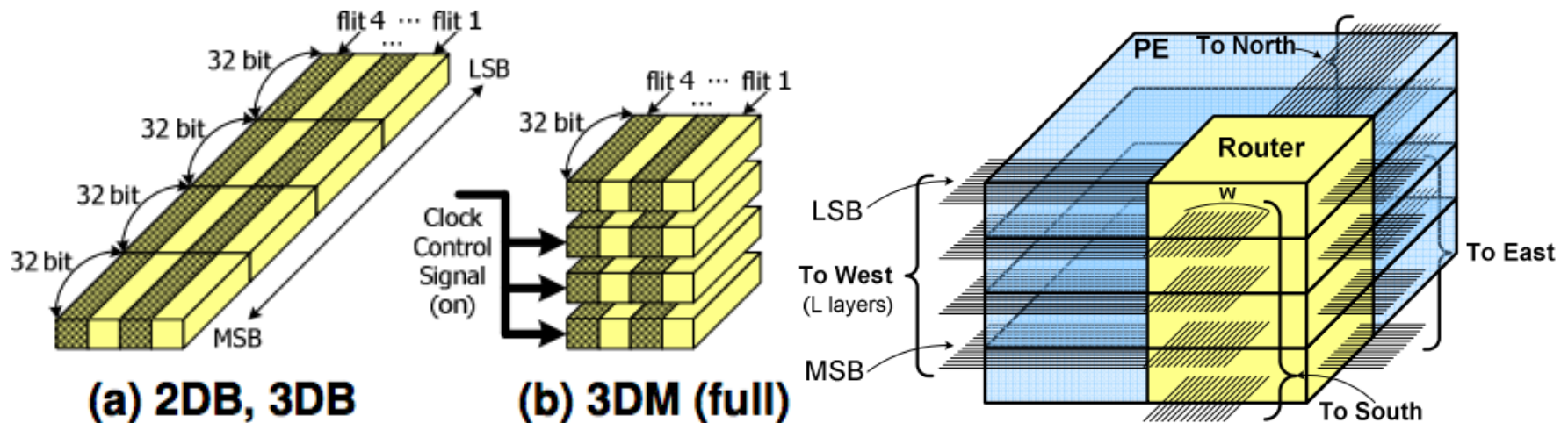◻ Reducing power by reducing voltage

◻ Data compression to lower dynamic power
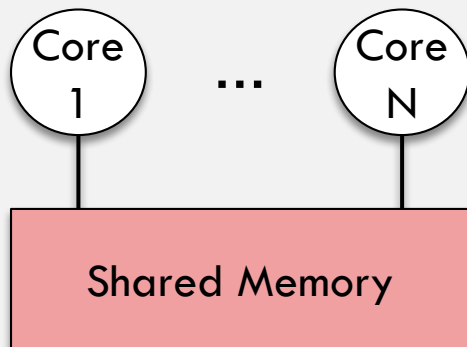
◻ Data encoding for reducing switching power

◻ etc.

Cache Coherence: Intro

Communication in Multiprocessors

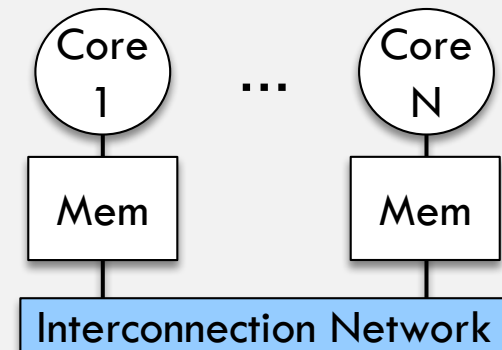☐ How multiple processor cores communicate?

Shared Memory	Message Passing
▪ Multiple threads employ shared memory ▪ Easy for programmers (loads and stores)	▪ Explicit communication through interconnection network ▪ Simple hardware

Shared Memory Architectures
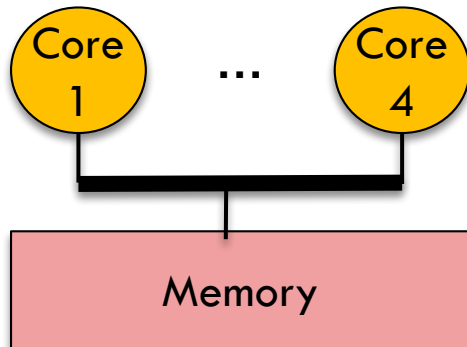
Uniform Memory Access	Non-Uniform Memory Access

- Equal latency for all processors
- Simple software control

- Access latency is proportional to proximity
  - Fast local accesses

Example UMA

Example NUMA
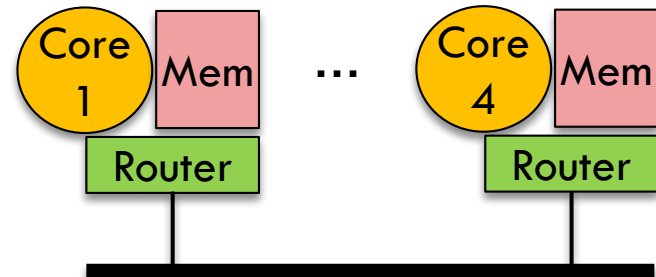
Network Topologies

Shared Network	Point to Point Network

- ☐ Low latency
- ☐ Low bandwidth
- ☐ Simple control
  - ▣ e.g., bus
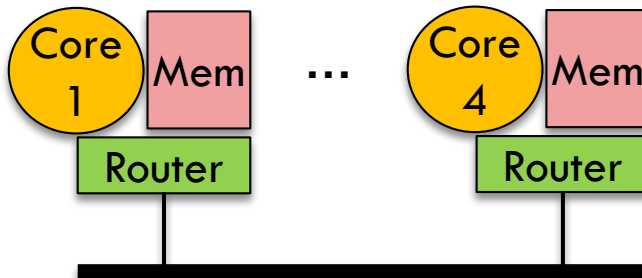
- ☐ High latency
- ☐ High bandwidth
- ☐ Complex control
  - ▣ e.g., mesh, ring
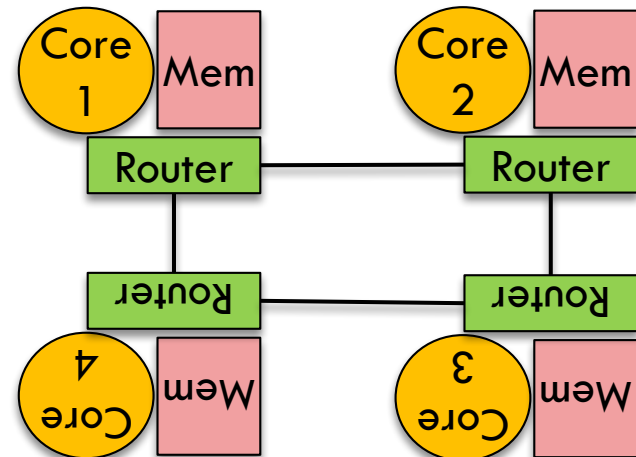
# Challenges in Shared Memories

□ Correctness of an application is influenced by

◻ Memory consistency

■ All memory instructions appear to execute in the program order

■ Known to the programmer

◻ Cache coherence

■ All the processors see the same data for a particular memory address as they should have if there were no caches in the system
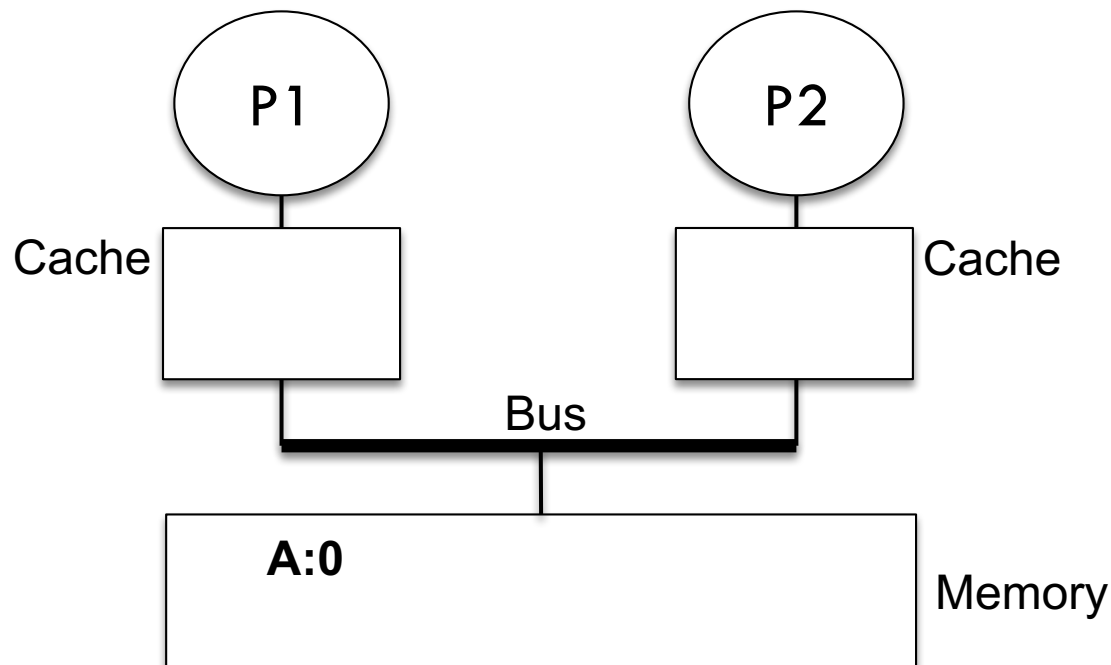
■ Invisible to the programmer

Cache Coherence Problem

□ Multiple copies of each cache block

  ◘ In main memory and caches

□ Multiple copies can get inconsistent when writes happen
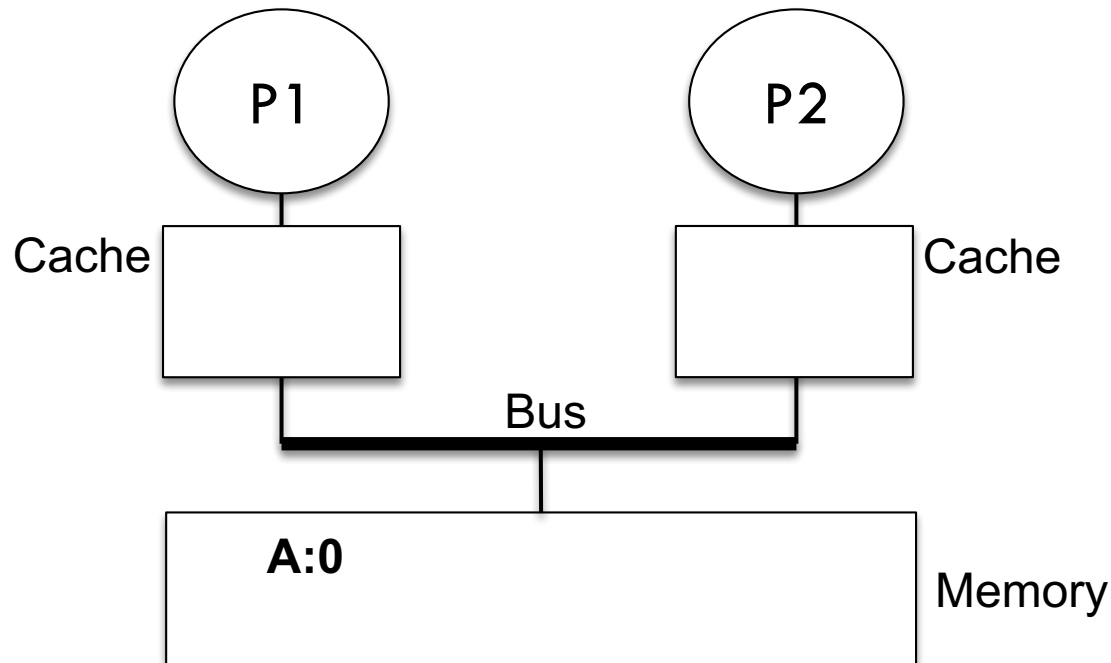
  ◘ Solution: propagate writes from one core to others
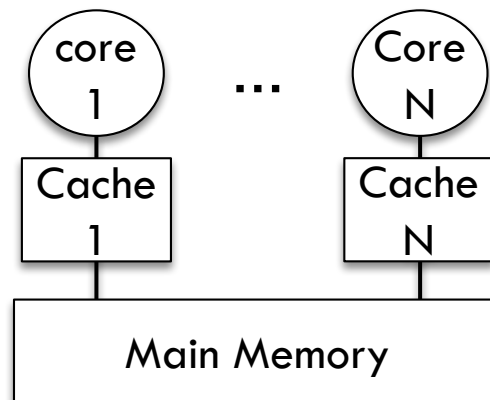
Scenario 1: Loading From Memory

- Variable A initially has value 0
- P1 stores value 1 into A
- P2 loads A from memory and sees old value 0

Scenario 2: Loading From Cache

- P1 and P2 both have variable A (value 0) in their caches

- P1 stores value 1 into A

- P2 loads A from its cache and sees old value

P1

P2

Cache

Cache

Bus

A:0

Memory

Cache Coherence

- The key operation is update/invalidate sent to all or a subset of the cores
  - Software based management
    - Flush: write all of the dirty blocks to memory
    - Invalidate: make all of the cache blocks invalid
  - Hardware based management
    - Update or invalidate other copies on every write
    - Send data to everyone, or only the ones who have a copy

- Invalidation based protocol is better. Why?