

Quiz 3 Review

Suvinay Subramanian

6.823 Spring 2016

Topics Snapshot

» Cache coherence

- Directory, snoopy protocols
- MSI, MESI, MOSI
- Synchronization

» Memory consistency models

- Sequential consistency
- Fences

» On-chip interconnection networks

- Metrics (diameter, bisection bandwidth, avg latency)
- Routing, flow control deadlock freedom
- Router microarchitecture

MULTI-CORE SYSTEMS!

Cache Coherence

- » Problem: If multiple processors cache the same block, how do they ensure “correct” view of the data?
- » Concerned with accesses to a single address.

Two rules:

1. Write propagation: Writes eventually become visible to other processors
2. Write serialization: Writes to same location are serialized

Coherence Protocols

» Invalidation vs Update-based

» Snoopy vs Directory-based

Coherence Protocols

» MSI states:

- Modified (M): Cache has exclusive access to line with read, write permissions.
- Shared (S): Cache has shared, read-only copy of line.
- Invalid (I): Cache does not have copy of data

» MSI optimizations:

- Exclusive (E) state
- Owned (O) state

Synchronization Primitives

» Why? Sequencing actions among multiple processes. Used to implement:

- Mutex (locks)
- Condition variables
- Semaphores

» Some synchronization primitives:

- Test-and-set
- Load-reserve, store-conditional
- Swap
- Compare-and-swap

Different primitives have different properties, and present different implementation tradeoffs.

Compare-and-swap

```
CAS old, new, Imm(base):  
    if (old == Mem[Imm+base])  
        Mem[Imm+base] ← new  
    else  
        old ← Mem[Imm+base]
```

» *Atomically* loads value at effective memory address, and compares value to value in register `old`.

- If both values are equal, update memory location with value in register `new`
- Else, update value in `old` with value in memory

Load-reserve, store-conditional

```
LR rs, (rt):  
    <flag, addr> ← <1, rt>  
    rs ← Memory[addr]
```

```
SC (rt), rs:  
    if <flag, addr> == <1, rt>:  
        Clear other procs flag  
        Mem[addr] ← rs  
        status ← 1  
    else:  
        status ← 0
```

See handout 13.
It shows you how to
implement a lock
using CAS.

L15, 16 show you
how to implement
locking using swap.

Different primitives
have different
properties, and
present different
implementation
tradeoffs.

Topics Snapshot

» Cache coherence

- Directory, snoopy protocols
- MSI, MESI, MOSI
- Synchronization

» Memory consistency models

- Sequential consistency
- Fences

» On-chip interconnection networks

- Metrics (diameter, bisection bandwidth, avg latency)
- Routing, flow control deadlock freedom
- Router microarchitecture

Memory Consistency Model

A memory (consistency) model specifies the order in which memory accesses performed by one thread become visible to other threads in the program.

» Contract between the hardware and software

» Loosely speaking, it specifies:

- Set of legal values a load can return
- Set of legal final memory states for a program

Memory Consistency Model

» Sequential Consistency (SC)

- Maintain program order
- Loads, stores appear atomic
- “Strongest”, most intuitive model

» Weak Memory Models

- Total Store Order (TSO)
- Partial Store Order (PSO)
- Relaxed Memory Order (RMO)



Enable several optimizations on the processor, memory system, interconnection network.

Memory Fences

- » Idea: Not all accesses need to be “strictly” ordered. Programmer identifies regions which need (not) be ordered.
- » Primitives that prevent otherwise permitted re-orderings of loads and stores
- » Different flavors on different systems:
 - Sparc: MEMBAR
 - x86: LFENCE, SFENCE, MFENCE

Topics Snapshot

» Cache coherence

- Directory, snoopy protocols
- MSI, MESI, MOSI
- Synchronization

» Memory consistency models

- Sequential consistency
- Fences

» On-chip interconnection networks

- Metrics (diameter, bisection bandwidth, avg latency)
- Routing, flow control deadlock freedom
- Router microarchitecture

Network-on-Chip

- » Handles communication between various on-chip elements: caches, memory-controller.

- » Several characteristics:
 - Topology
 - Flow control
 - Routing
 - Router Microarchitecture

Topology

» Arrangement of channels and nodes i.e. how different nodes connect to each other.

- Ring, mesh, torus, tree etc.

» Properties

- Diameter
- Average distance / average latency
- Bisection bandwidth

Routing

» Path from source to destination

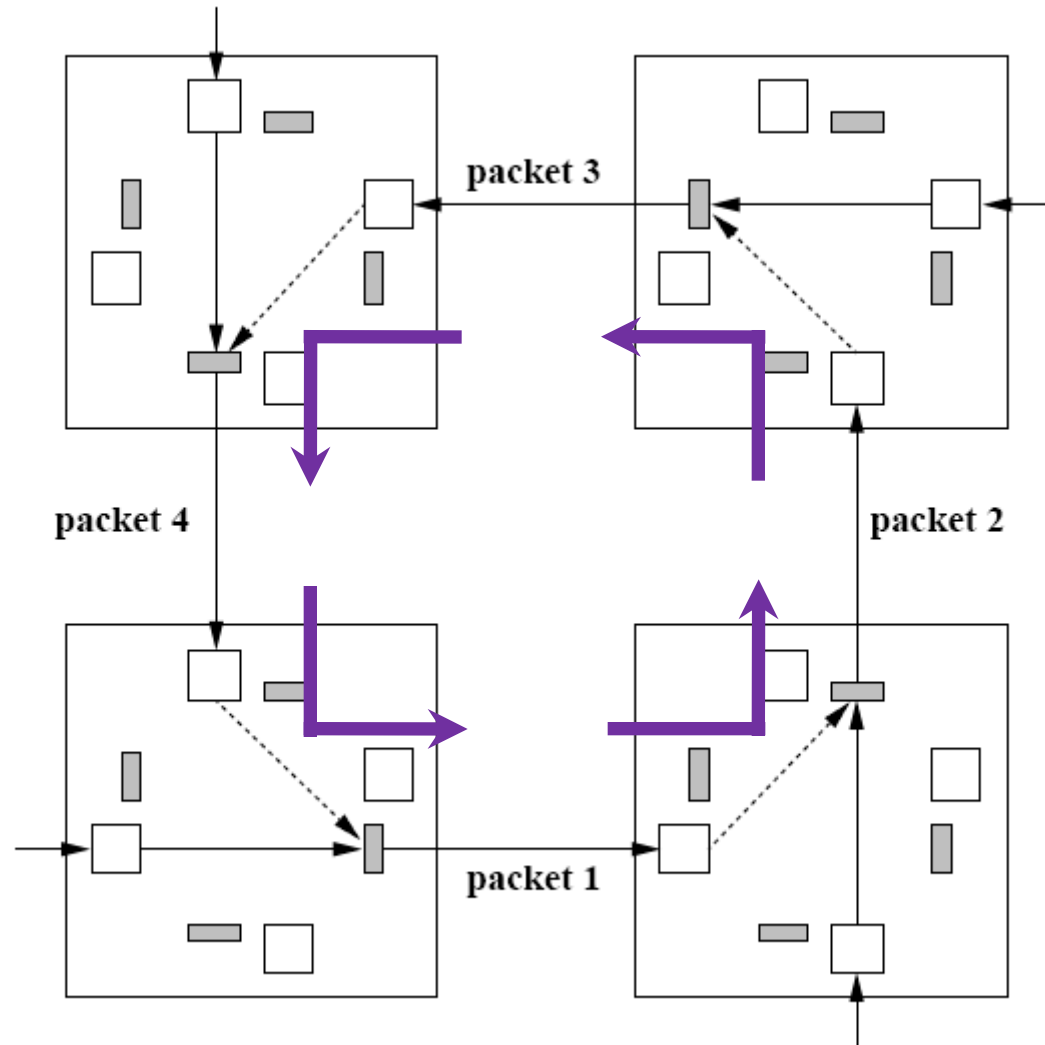
» Properties:

- Deterministic/oblivious
- Adaptive
- Minimal
- Balanced
- Deadlock-free

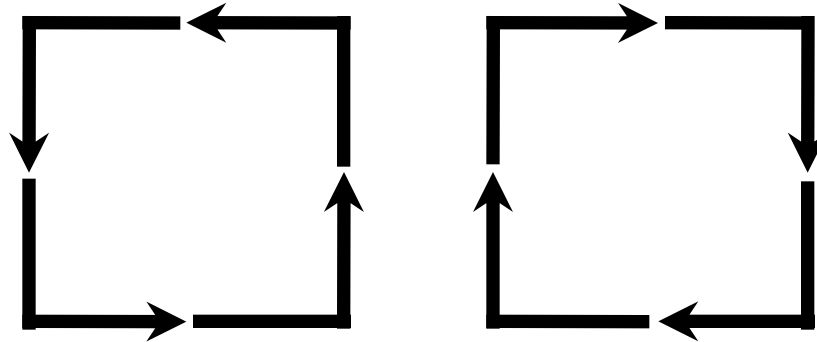


Some of these
are often at odds
with each other.

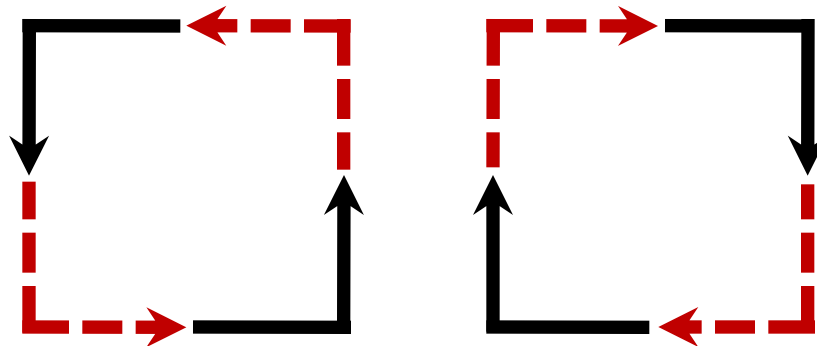
Routing Deadlock



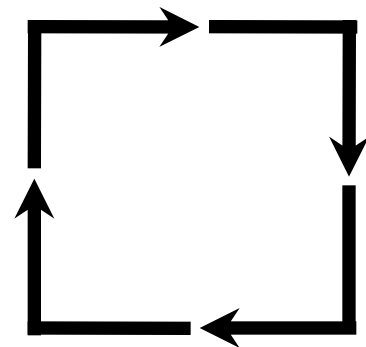
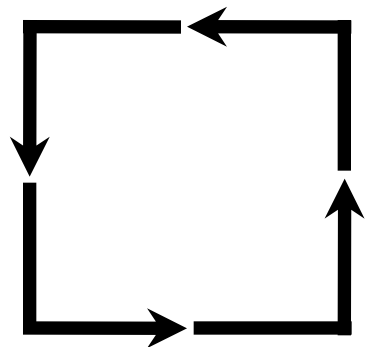
Turn Model



The eight possible turns and cycles in a 2D mesh



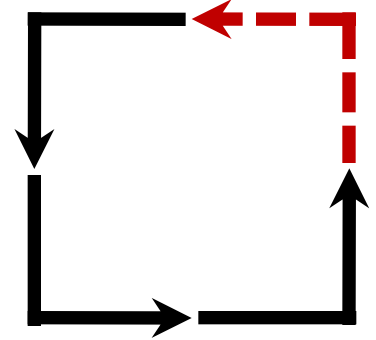
Only four turns are allowed in the XY routing algorithm



All possible turns

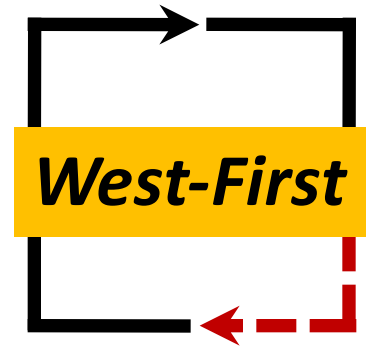
Restrict 1 turn

Restrict 1 turn

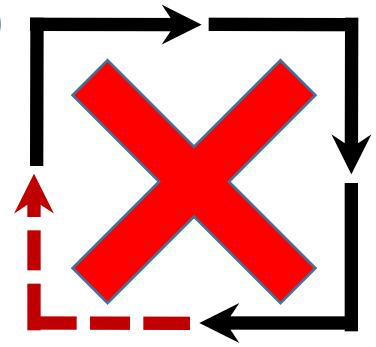


AND

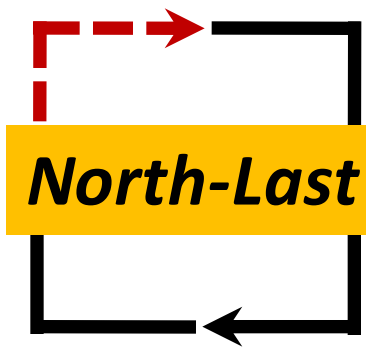
1



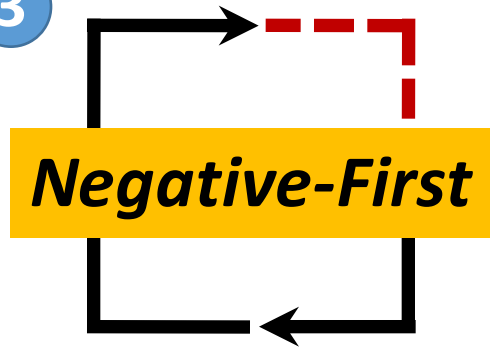
4



2

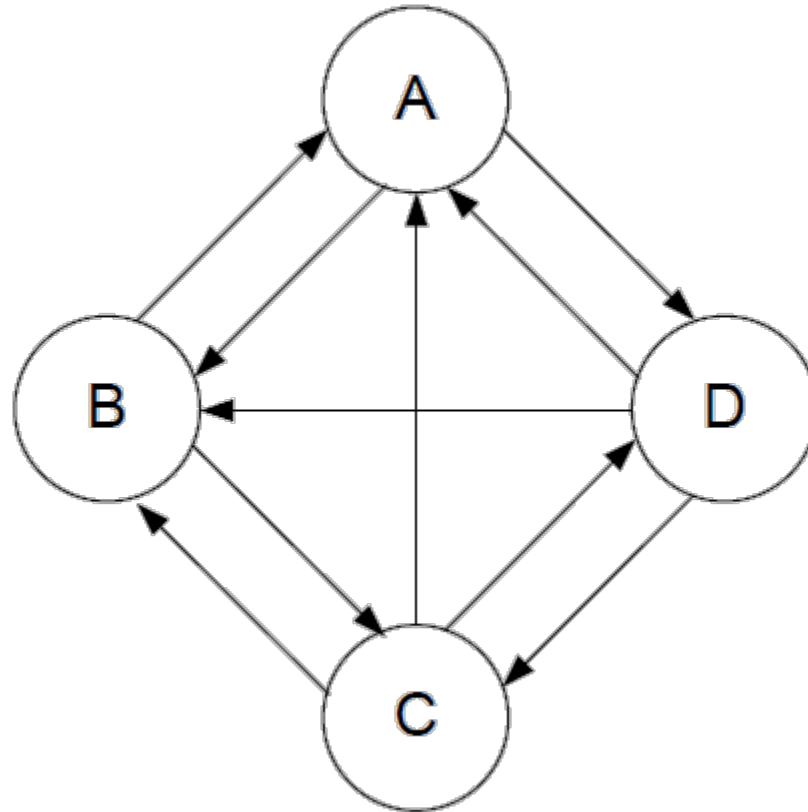


3



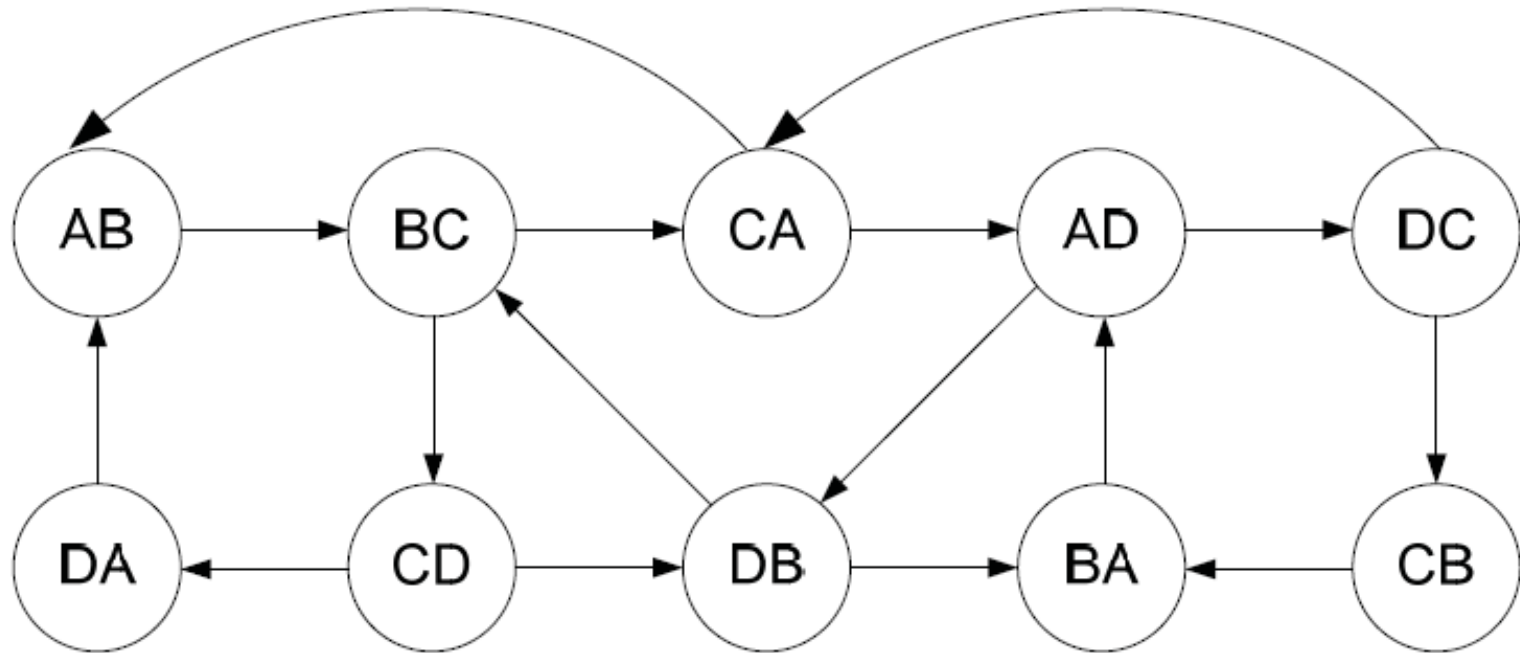
Channel Dependency Graph (CDG)

Cycles in the CDG point to potential deadlock.



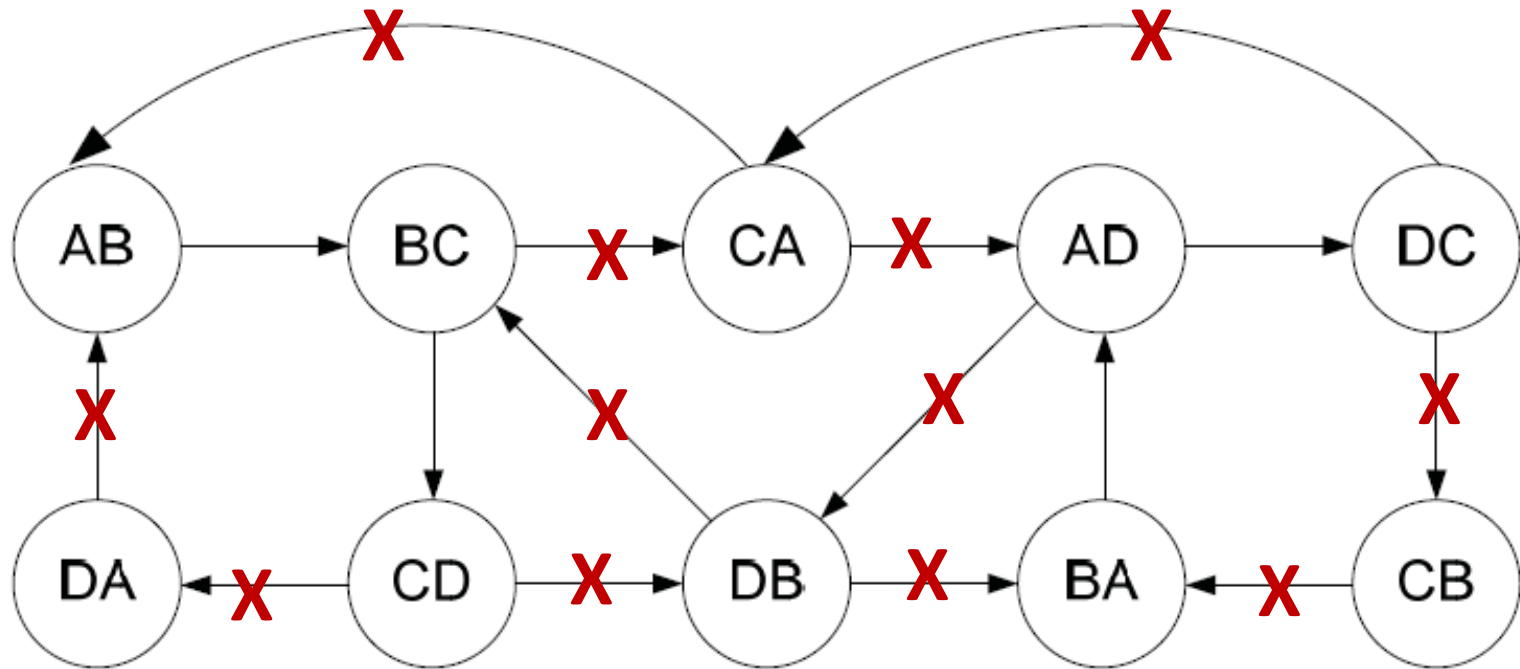
» Vertices in the CDG represent network links. Edges represent if a particular turn is allowed.

Channel Dependency Graph (CDG)



Deadlock free? No

Channel Dependency Graph (CDG)



Minimal routing

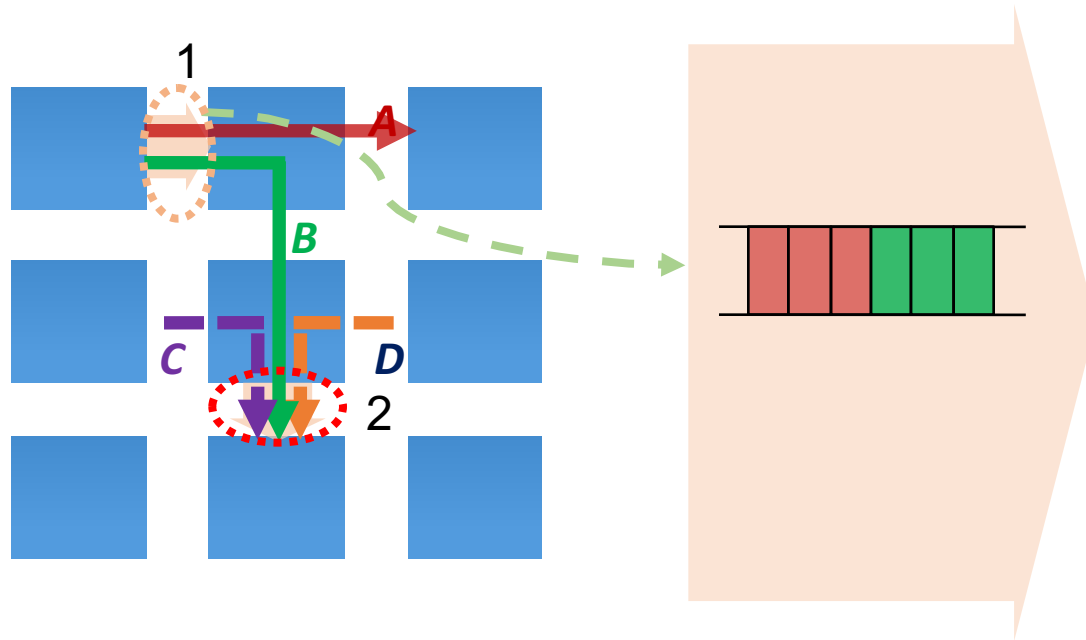
Deadlock free? Yes

Flow Control

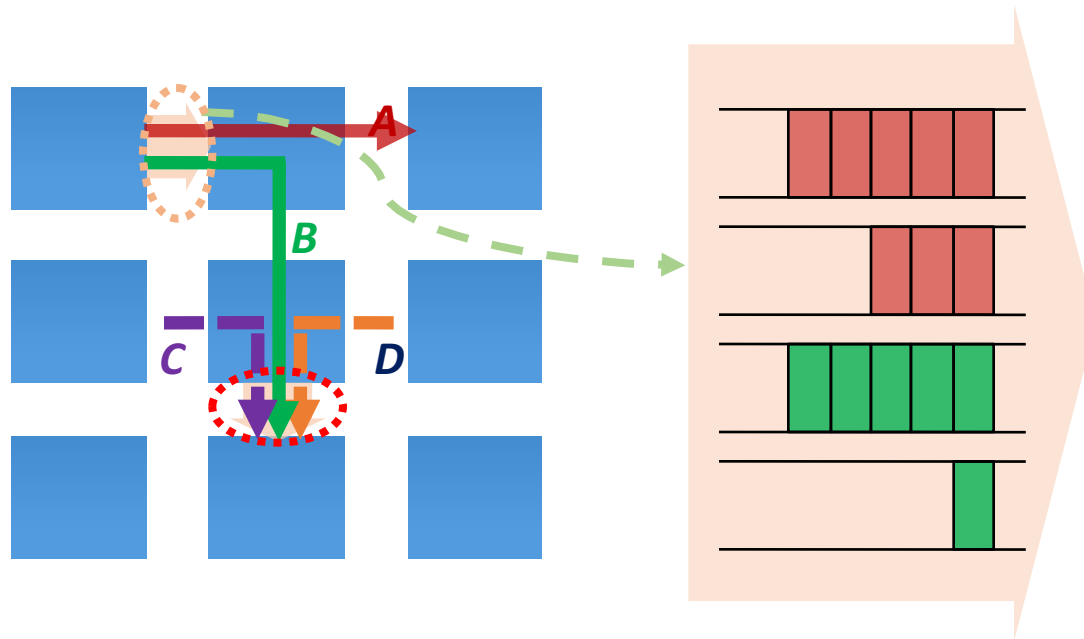
- » How network resources are allocated to packets traversing the network

- » Bufferless
 - Circuit switching, dropping, mis-routing
- » Buffered
 - Store-and-forward, virtual cut-through, wormhole, virtual-channel

Head-of-line (HoL) Blocking

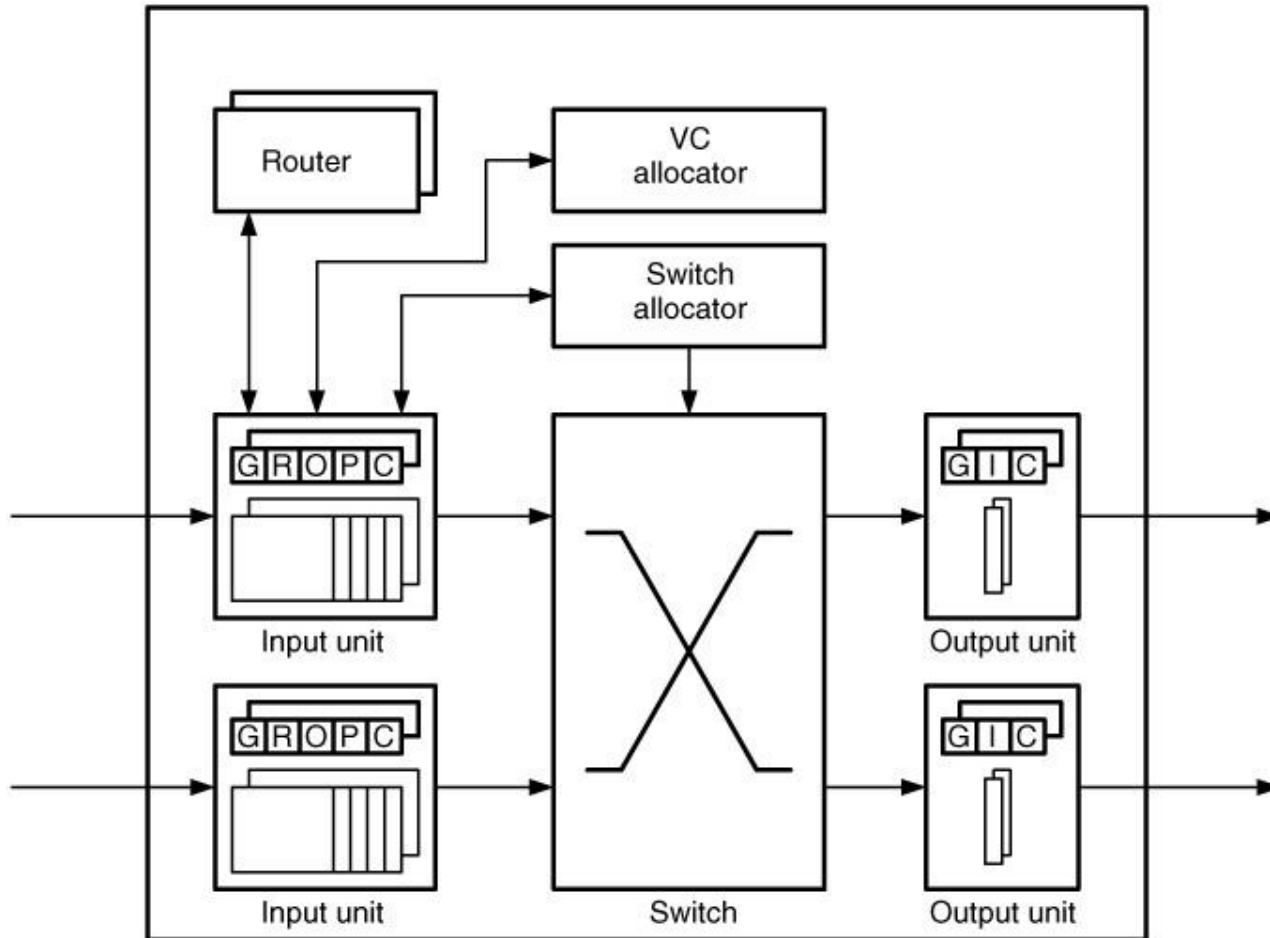


Head-of-line (HoL) Blocking



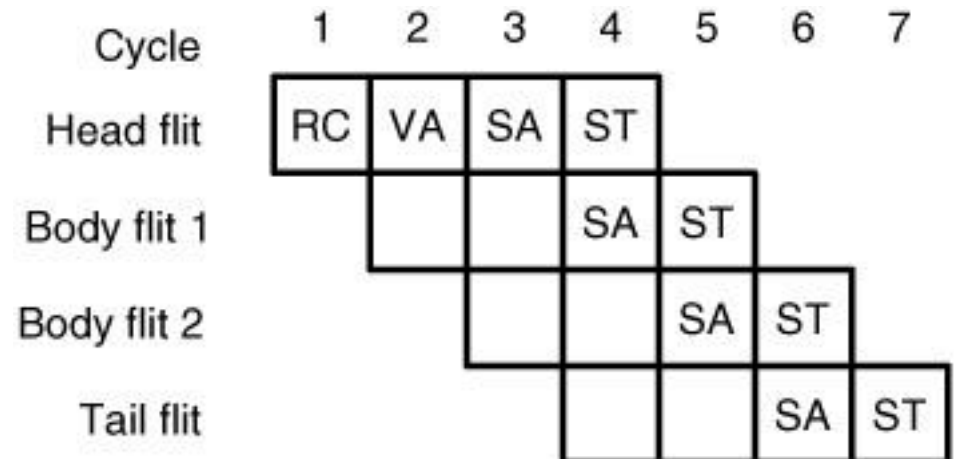
Solution: Virtual Channels

Router Microarchitecture



Router Pipeline

1. Buffer Write (BW)
2. Route Compute (RC)
3. Virtual Channel Allocation (VA)
4. Switch Allocation (SA)
5. Switch Traversal (ST)
6. Link Traversal (LT)



All the best! 😊