On-Chip Networks II: Router Microarchitecture & Routing

Daniel Sanchez
Computer Science & Artificial Intelligence Lab
M.I.T.

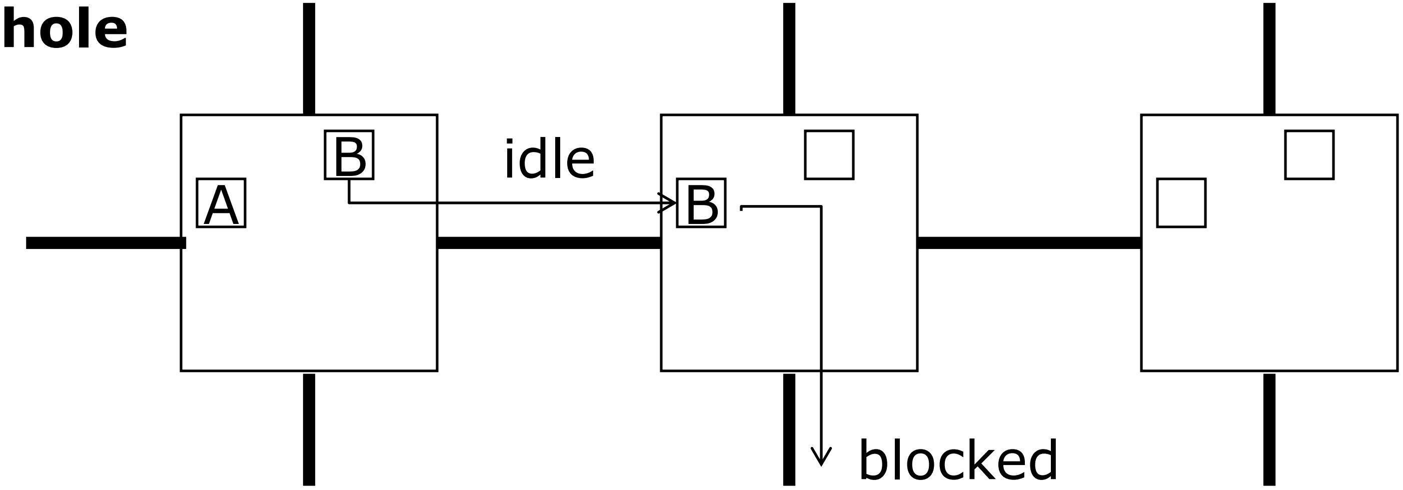
Recap: Wormhole Flow Control

- Each router manages buffers in flits
- Each packet is sent through output link as soon as possible (without waiting for all its flits to arrive)
- Router buffers are not large enough to hold full packet → on congestion, packet's flits often buffered across routers
- Problem: On congestion, links assigned to a blocked packet cannot be used by other packets

Recap: Wormhole Flow Control

- Each router manages buffers in flits
- Each packet is sent through output link as soon as possible (without waiting for all its flits to arrive)
- Router buffers are not large enough to hold full packet → on congestion, packet's flits often buffered across routers
- Problem: On congestion, links assigned to a blocked packet cannot be used by other packets

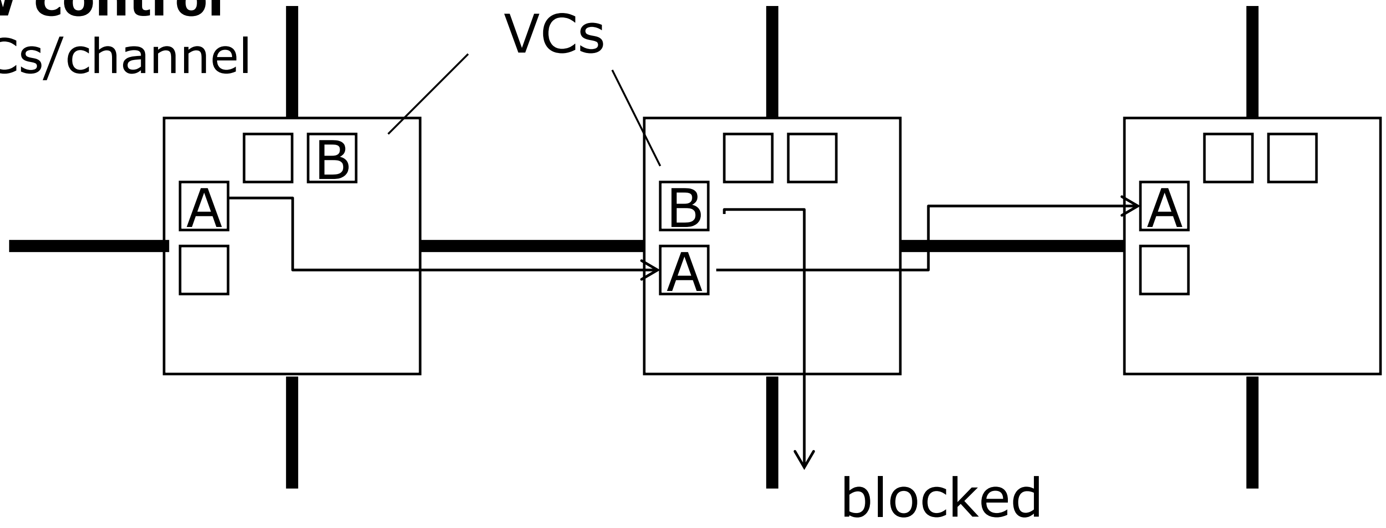
Wormhole



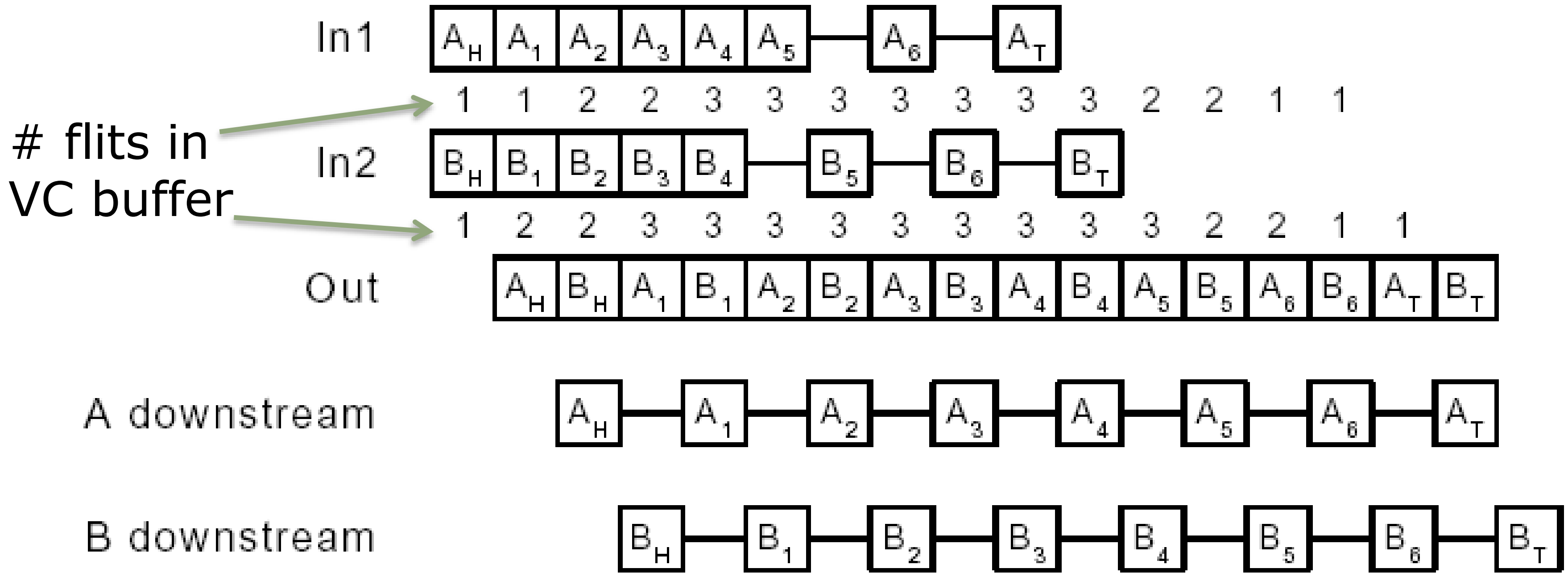
Recap: Virtual-Channel Flow Control

- When a packet blocks, instead of holding on to channel, hold on to virtual channel
- Virtual channel (VC) = channel state + flit buffers
- Multiple virtual channels reduce blocking
- Ex: Wormhole (=1 VC/channel) vs 2 VCs/channel

VC flow control
with 2 VCs/channel



Time-Space View: Virtual-Channel



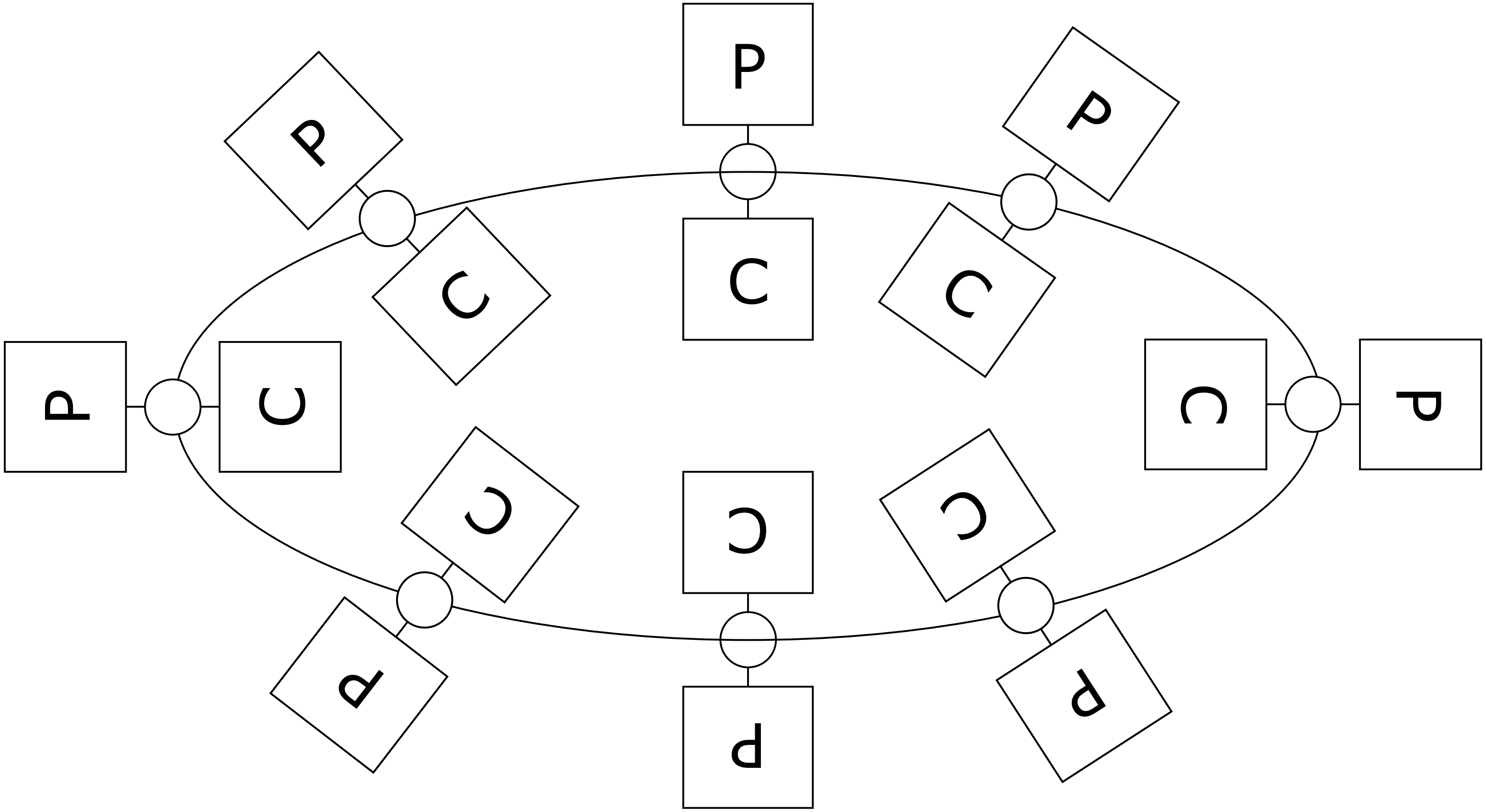
In1
A_H A_1 A_2 A_3 A_4 A_5 A_6 A_T
1 1 2 2 3 3 3 3 3 3 3 2 2 1 1
# flits in VC buffer
In2
B_H B_1 B_2 B_3 B_4 B_5 B_6 B_T
1 2 2 3 3 3 3 3 3 3 3 3 2 2 1 1
Out
A_H B_H A_1 B_1 A_2 B_2 A_3 B_3 A_4 B_4 A_5 B_5 A_6 B_6 A_T B_T
A downstream
A_H A_1 A_2 A_3 A_4 A_5 A_6 A_T
B downstream
B_H B_1 B_2 B_3 B_4 B_5 B_6 B_T

Interconnection Network Architecture

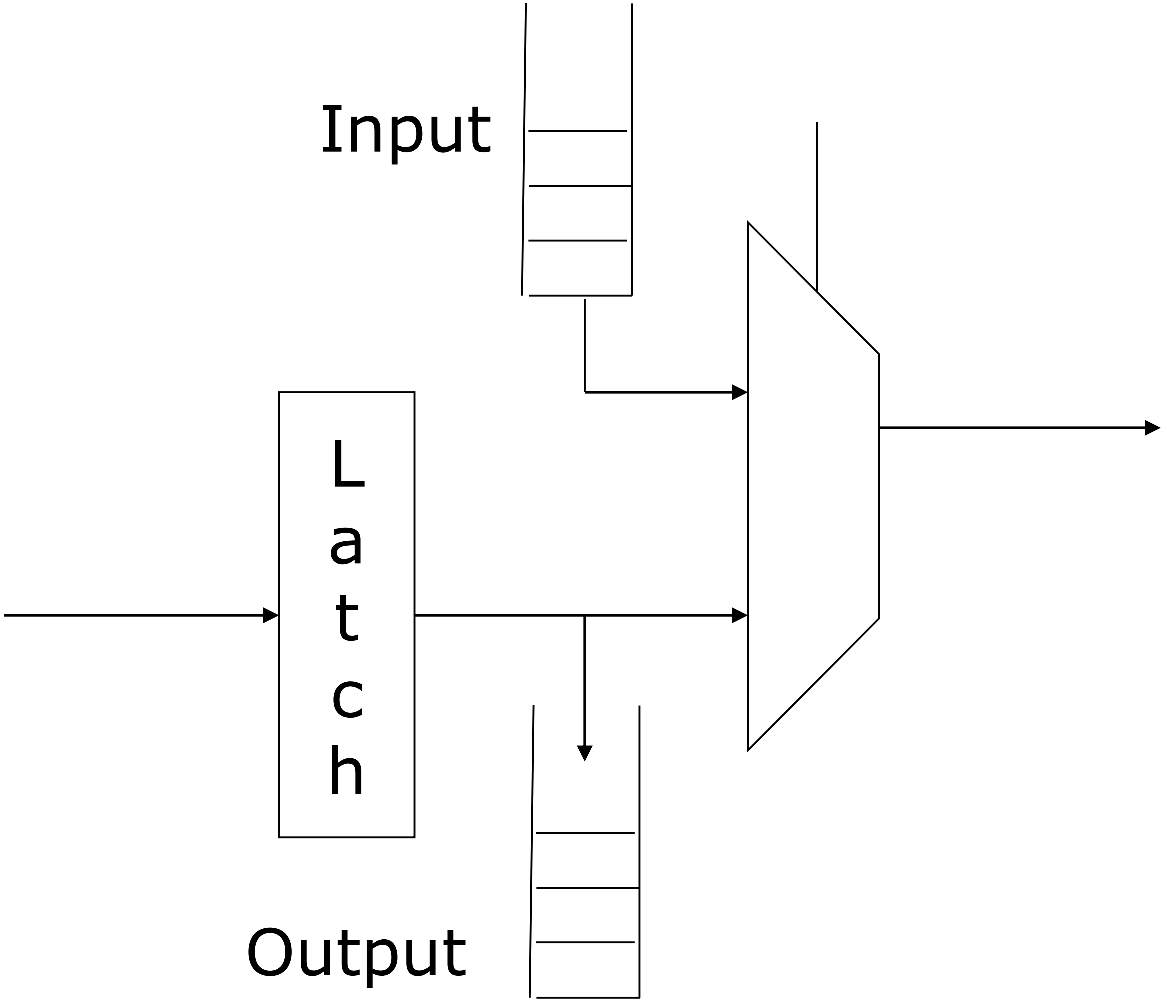
- *Topology*: How to connect the nodes up? (processors, memories, router line cards, …)
- *Routing*: Which path should a message take?
- *Flow control*: How is the message actually forwarded from source to destination?
- *Router microarchitecture*: How to build the routers?
- *Link microarchitecture*: How to build the links?

Router Microarchitecture

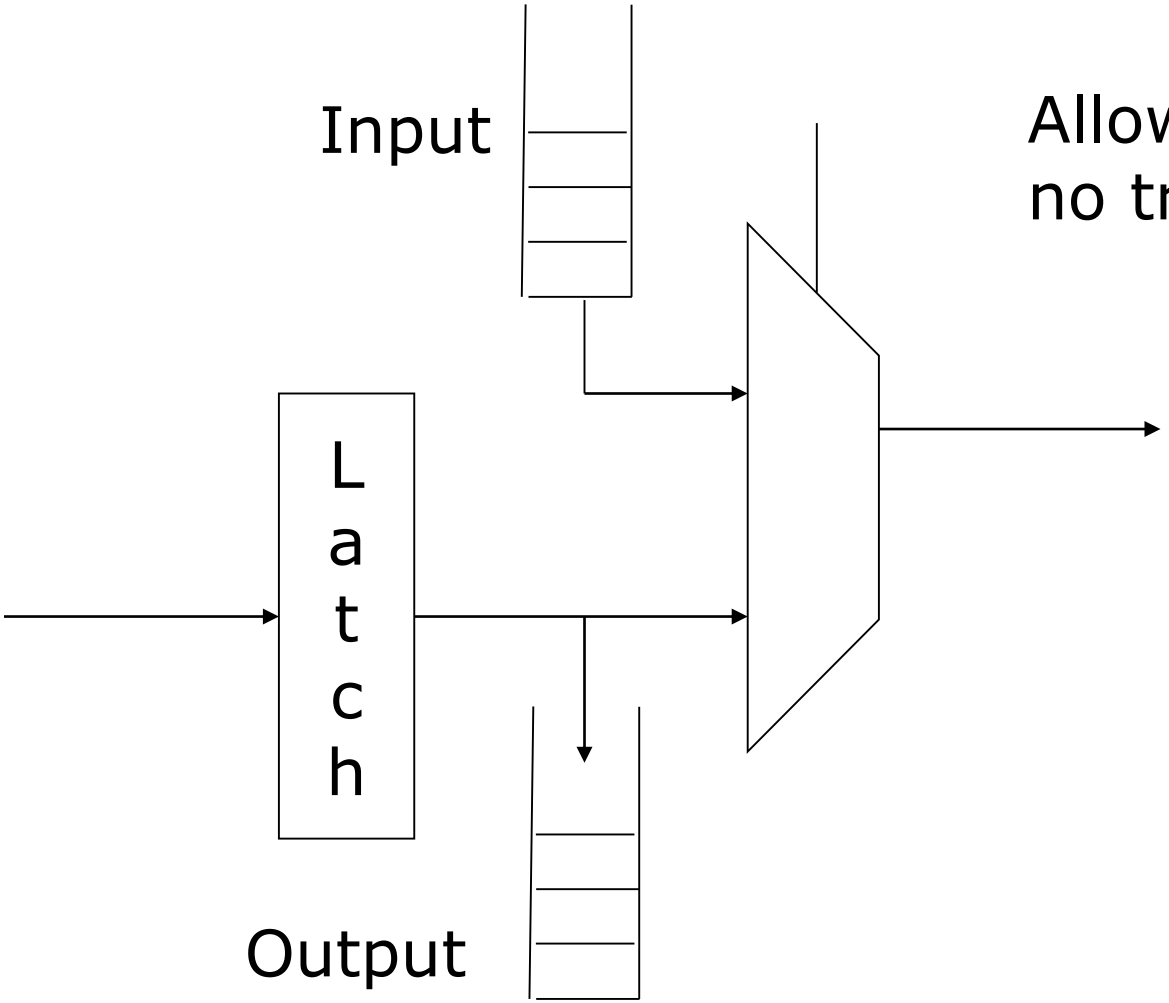
Ring-based Interconnect



Ring Stop

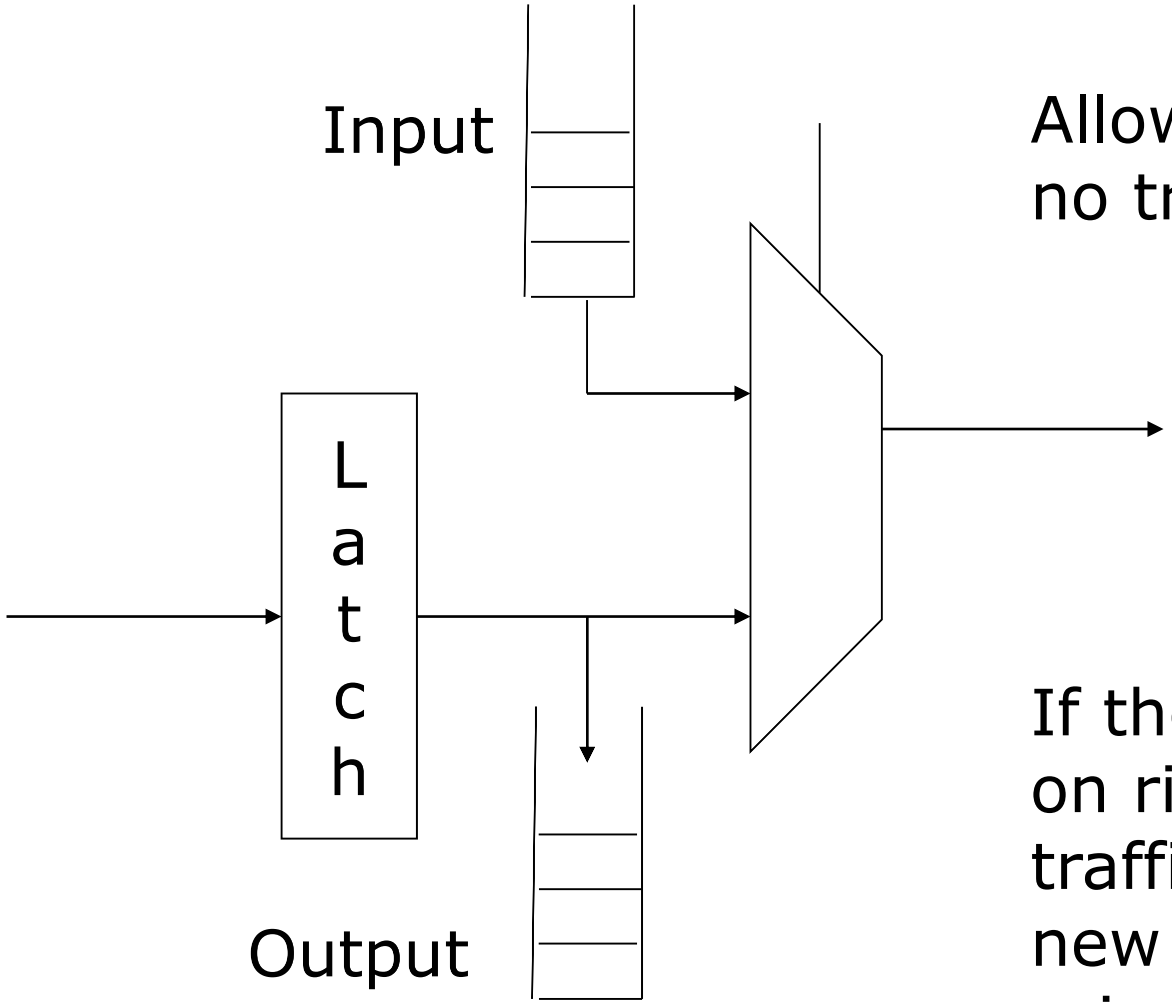


Ring Stop



Allow input if
no traffic on ring

Ring Stop



Allow input if no traffic on ring

If there is traffic on ring, should traffic on ring or new input get priority?

Ring Flow Control: Priorities

Rotary Rule – traffic in ring has priority

Ring Flow Control: Bounces

- What if traffic on the ring cannot get delivered, e.g., if output FIFO is full?

Ring Flow Control: Bounces

- What if traffic on the ring cannot get delivered, e.g., if output FIFO is full?

- One alternative: Continue on ring (bounce)

Ring Flow Control: Bounces

- What if traffic on the ring cannot get delivered, e.g., if output FIFO is full?

- One alternative: Continue on ring (bounce)

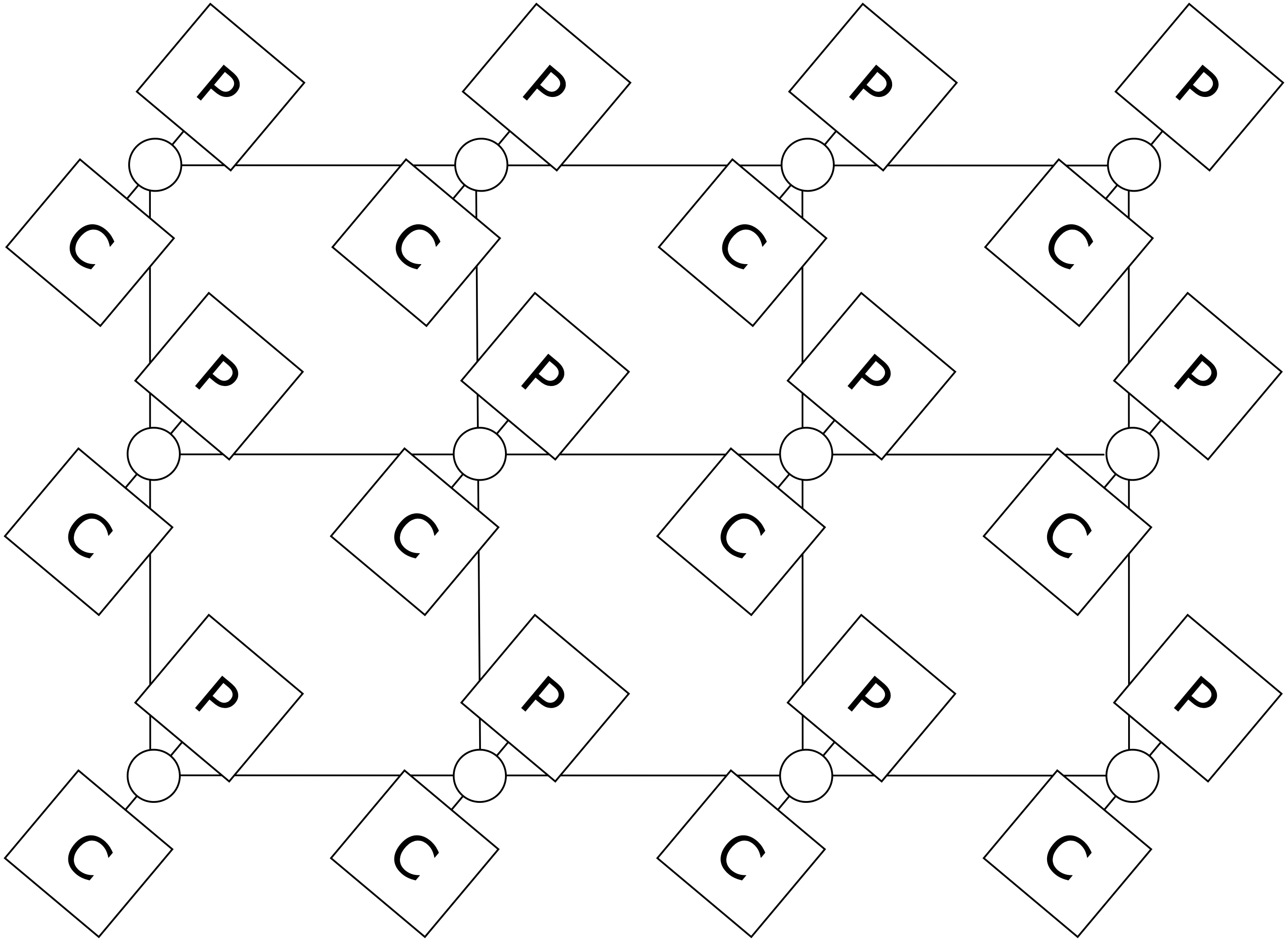
- *What are the consequences of such bounces?*

Ring Flow Control: Bounces

- What if traffic on the ring cannot get delivered, e.g., if output FIFO is full?
- One alternative: Continue on ring (bounce)
- *What are the consequences of such bounces?*

Traffic on ring no longer FIFO

General Interconnect
Tilera, Knights Landing…



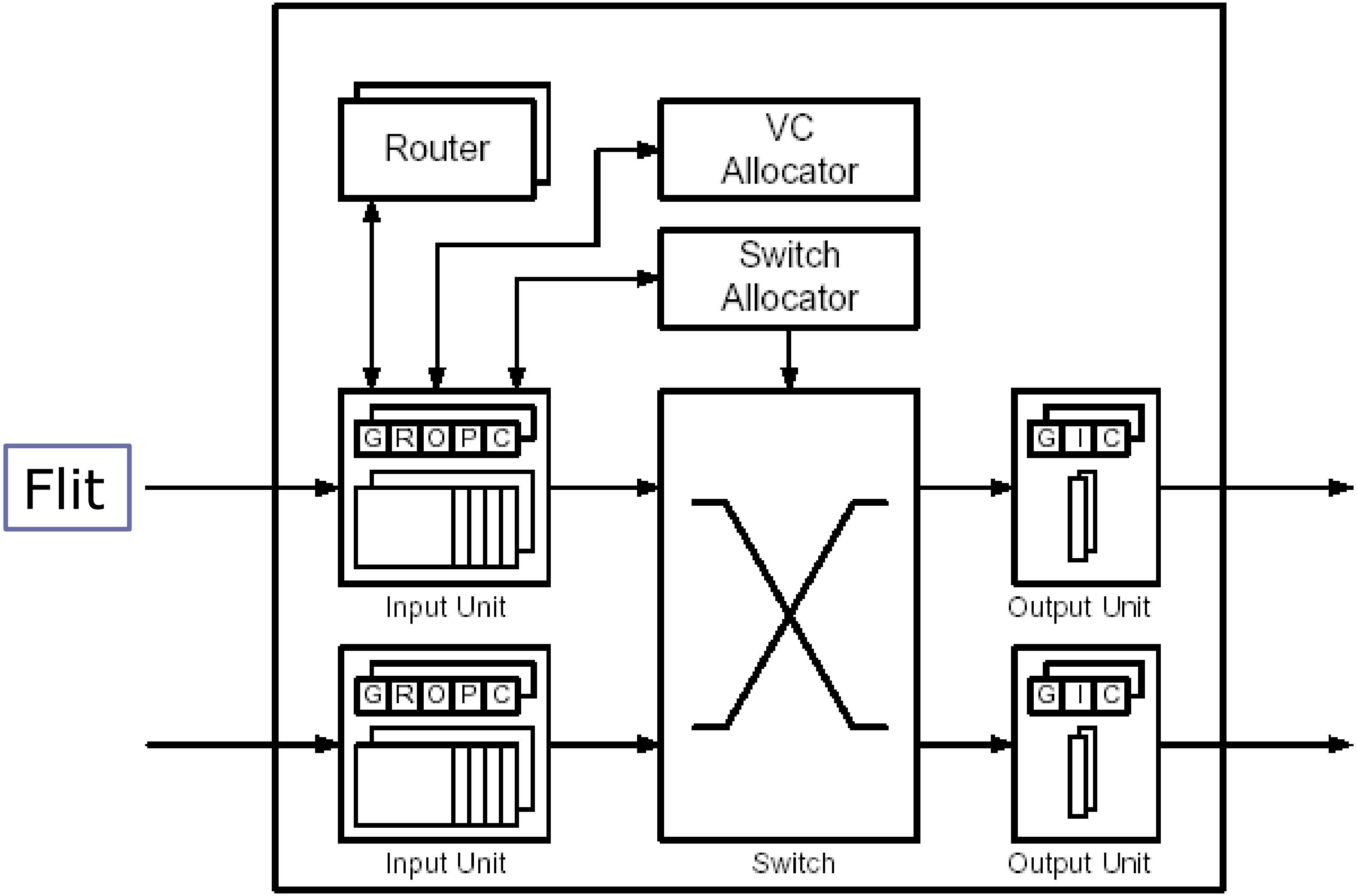
What’s In A Router?

- It’s a system as well

What’s In A Router?

- It’s a system as well
  - Logic – State machines, Arbiters, Allocators
    - Control data movement through router
    - Idle, Routing, Waiting for resources, Active
  - Memory – Buffers
    - Store flits before forwarding them
    - SRAMs, registers, processor memory
  - Communication – Switches
    - Transfer flits from input to output ports
    - Crossbars, multiple crossbars, fully-connected, bus

Virtual-channel Router



Router Pipeline vs. Processor Pipeline

- Logical stages:
  - BW
  - RC
  - VA
  - SA
  - BR
  - ST
  - LT
- Different flits go through different stages
- Different routers have different variants
  - E.g. speculation, lookaheads, bypassing
- Different implementations of each pipeline stage

- Logical stages:
  - IF
  - ID
  - EX
  - MEM
  - WB
- Different instructions go through different stages
- Different processors have different variants
  - E.g. speculation, ISA
- Different implementations of each pipeline stage

Baseline Router Pipeline

Head	BW	RC	VA	SA	ST	LT			
Body 1		BW	☒	☒	SA	ST	LT		
Body 2			BW	☒	☒	SA	ST	LT	
Tail				BW	☒	☒	SA	ST	LT

- Route computation performed once per packet
- Virtual channel allocated once per packet
- Body and tail flits inherit this info from head flit

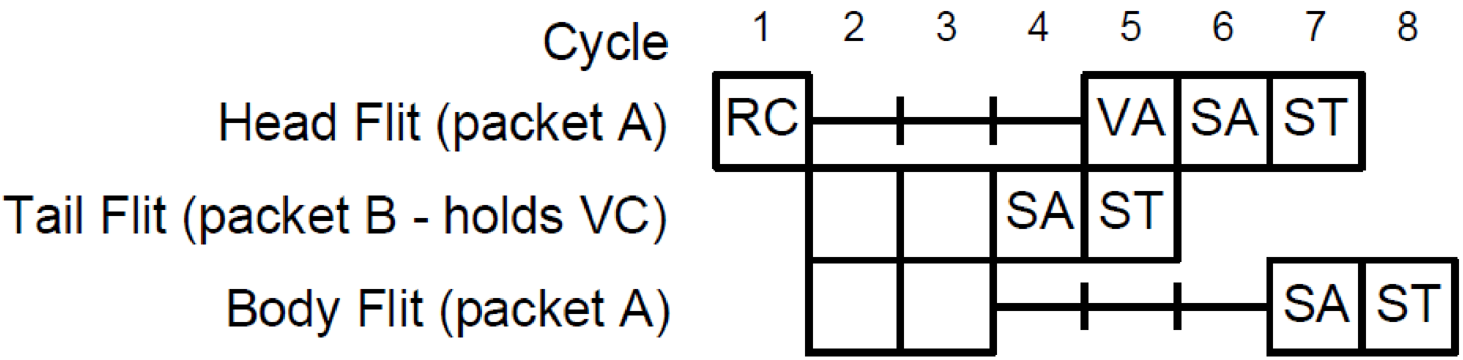
Allocators In Routers

- VC Allocator
  - Input VCs requesting for a range of output VCs
  - Example: A packet of VC0 arrives at East input port. It’s destined for west output port, and would like to get any of the VCs of that output port.
- Switch Allocator
  - Input VCs of an input port request for different output ports (e.g., One’s going North, another’s going West)
- “Greedy” algorithms used for efficiency

Allocators In Routers

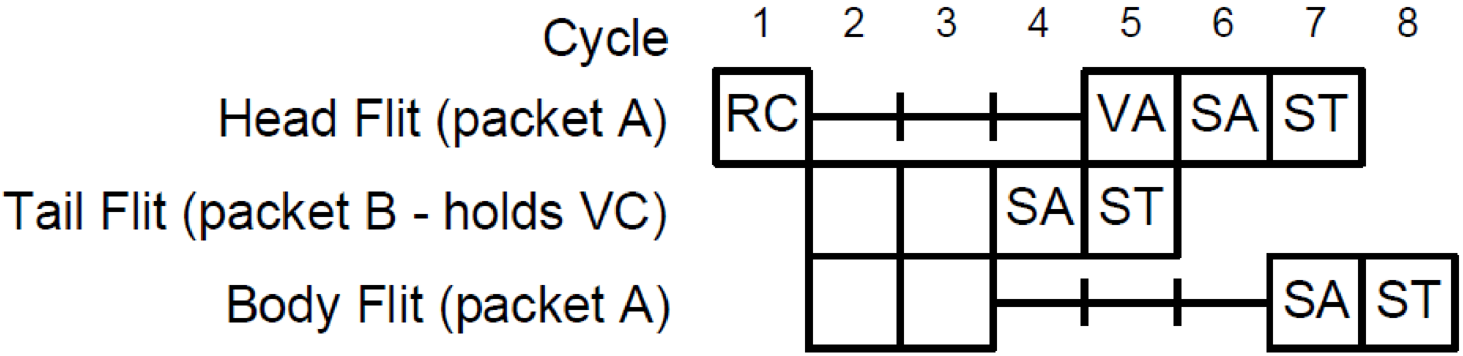
- VC Allocator
  - Input VCs requesting for a range of output VCs
  - Example: A packet of VC0 arrives at East input port. It's destined for west output port, and would like to get any of the VCs of that output port.
- Switch Allocator
  - Input VCs of an input port request for different output ports (e.g., One's going North, another's going West)
- "Greedy" algorithms used for efficiency
- What happens if allocation fails on a given cycle?

VC & Switch Allocation Stalls

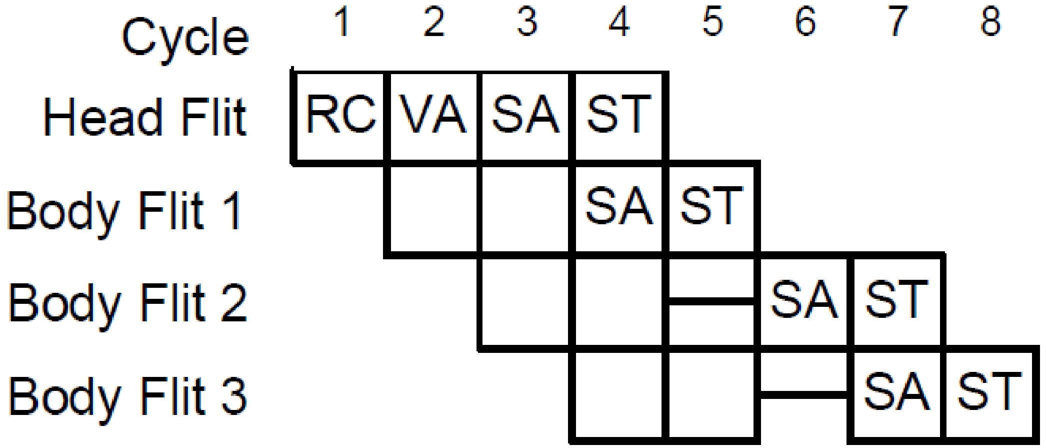


Cycle	1	2	3	4	5	6	7	8
Head Flit (packet A)	RC				VA	SA	ST	
Tail Flit (packet B - holds VC)				SA	ST			
Body Flit (packet A)							SA	ST

VC & Switch Allocation Stalls



Cycle	1	2	3	4	5	6	7	8
Head Flit (packet A)	RC	—	—	—	VA	SA	ST	
Tail Flit (packet B - holds VC)				SA	ST			
Body Flit (packet A)				—	—	—	SA	ST



Cycle	1	2	3	4	5	6	7	8
Head Flit	RC	VA	SA	ST				
Body Flit 1				SA	ST			
Body Flit 2					—	SA	ST	
Body Flit 3						—	SA	ST

Pipeline Optimizations: Lookahead Routing [Galles, SGI Spider Chip]

- At current router, perform route computation for next router

BW RC	VA NRC	SA	ST	LT

  - Head flit already carries output port for next router
  - RC just has to read output → fast, can be overlapped with BW
  - Precomputing route allows flits to compete for VCs immediately after BW
  - Routing computation for the next hop (NRC) can be computed in parallel with VA

- Or simplify RC (e.g., X-Y routing is very fast)

Pipeline Optimizations: Speculative Switch Allocation [Peh & Dally, 2001]

- Assume that Virtual Channel Allocation stage will be successful
  - Valid under low to moderate loads
- If both successful, VA and SA are done in parallel

BW RC	VA SA	ST	LT

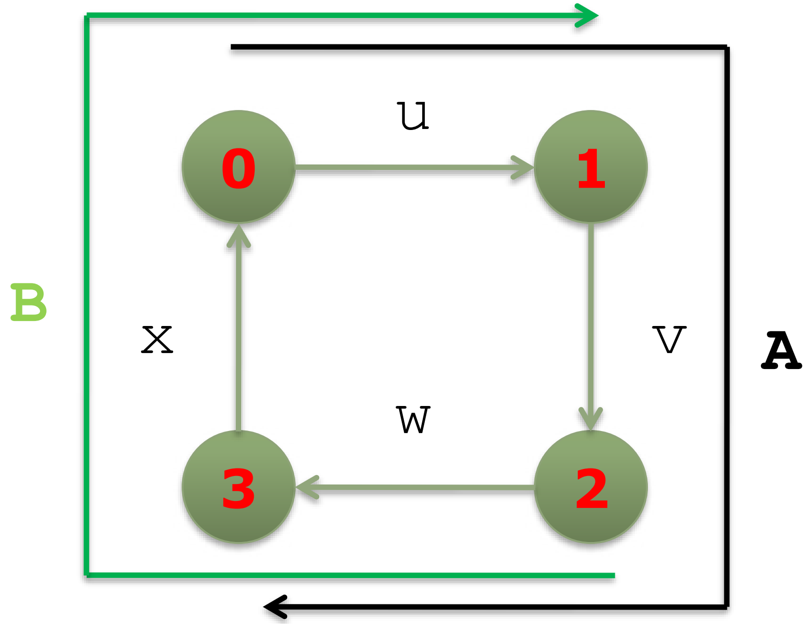
- If VA unsuccessful (no virtual channel returned)
  - Must repeat VA/SA in next cycle
- Prioritize non-speculative SA requests

Routing

Properties of Routing Algorithms

- Deterministic/Oblivious
  - 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 no packets move forward

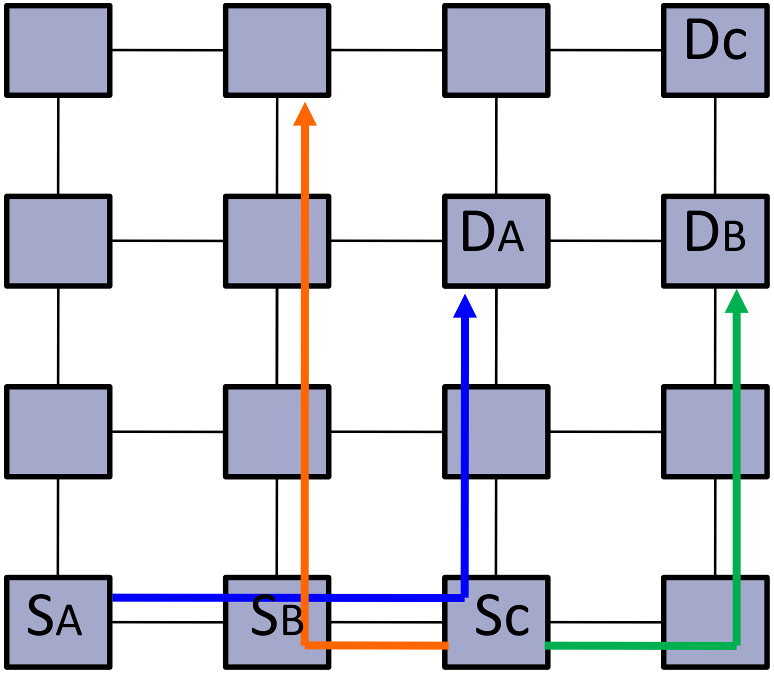
Network Deadlock



- Flow A holds u and v but cannot make progress until it acquires channel w
- Flow B holds channels w and x but cannot make progress until it acquires channel u

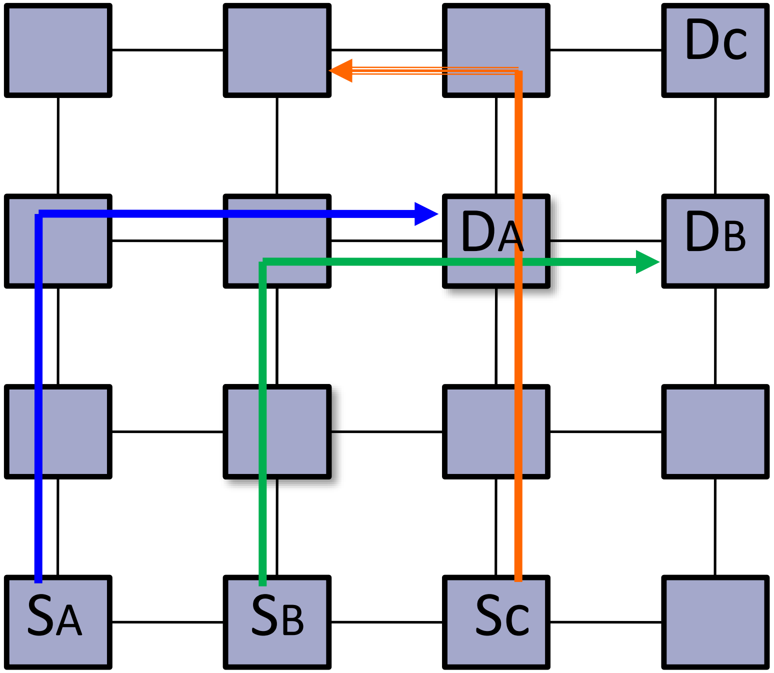
Dimension-Order Routing

XY-order



Uses 2 out of 4 turns

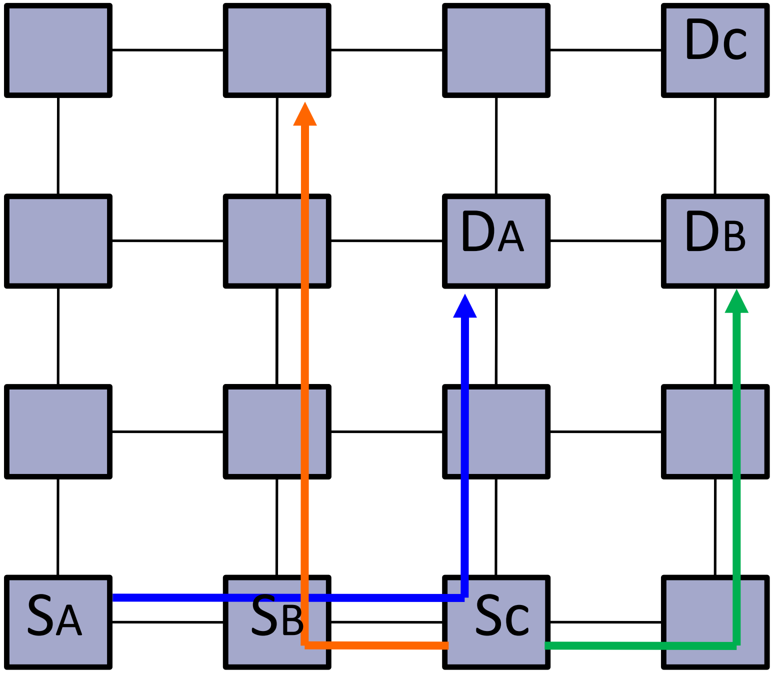
YX-order



Uses 2 out of 4 turns

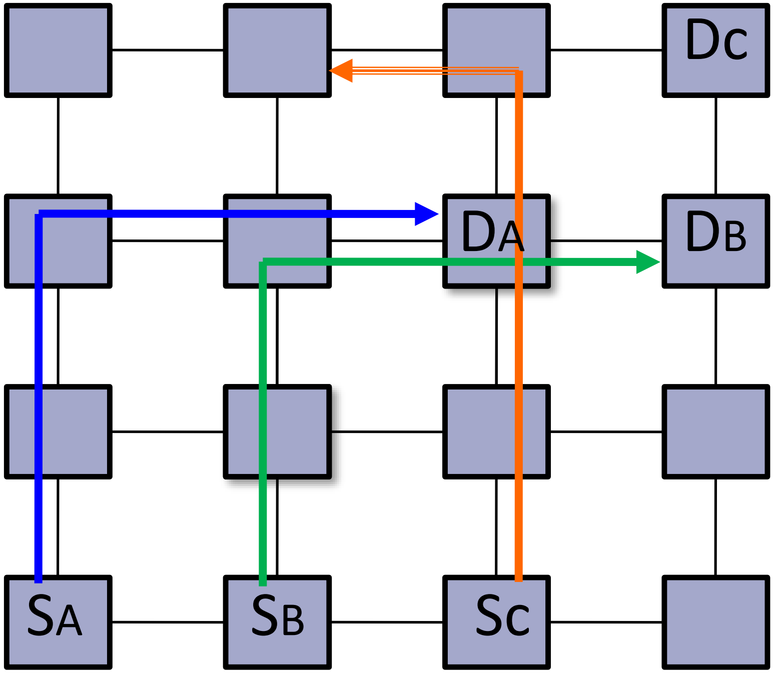
Dimension-Order Routing

XY-order



Uses 2 out of 4 turns

YX-order

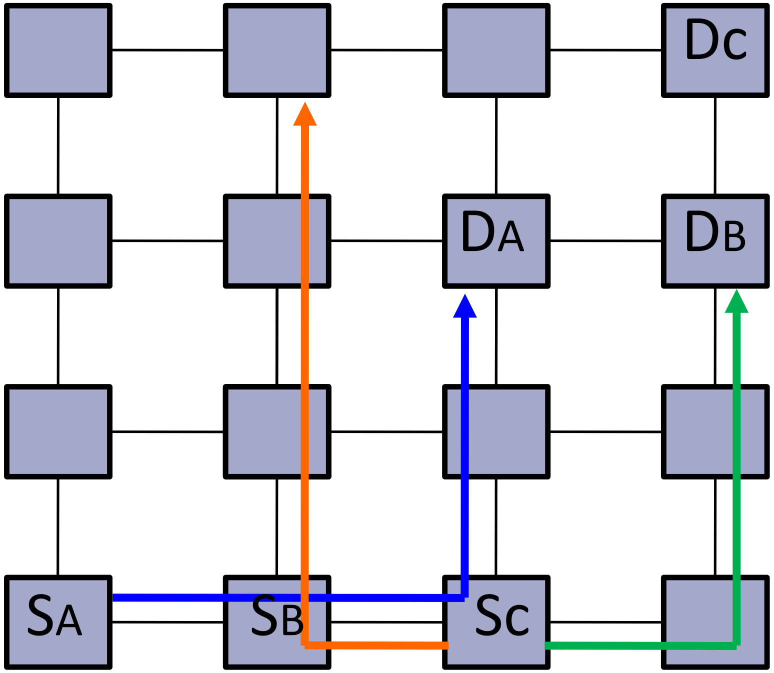


Uses 2 out of 4 turns

XY is deadlock free, YX is deadlock free, what about XY+YX?

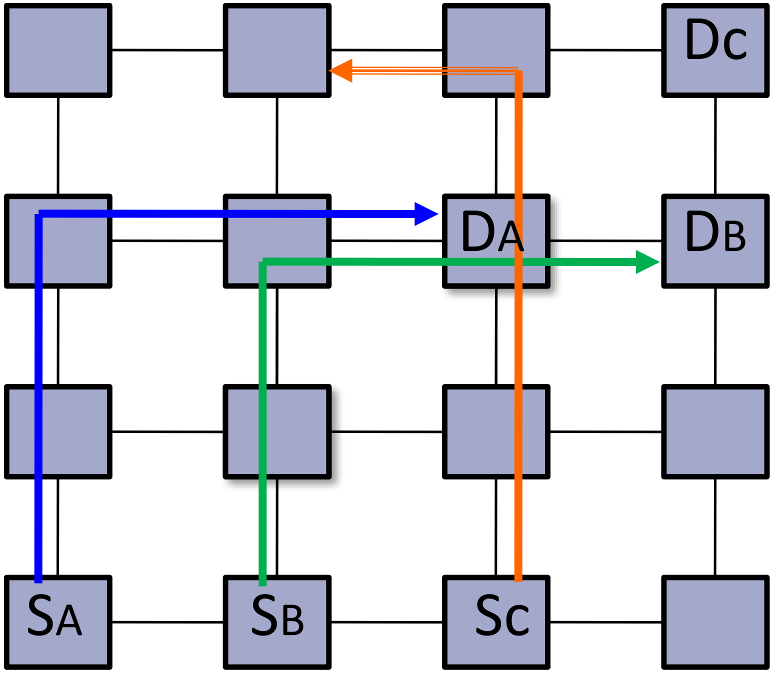
Dimension-Order Routing

XY-order



Uses 2 out of 4 turns

YX-order



Uses 2 out of 4 turns

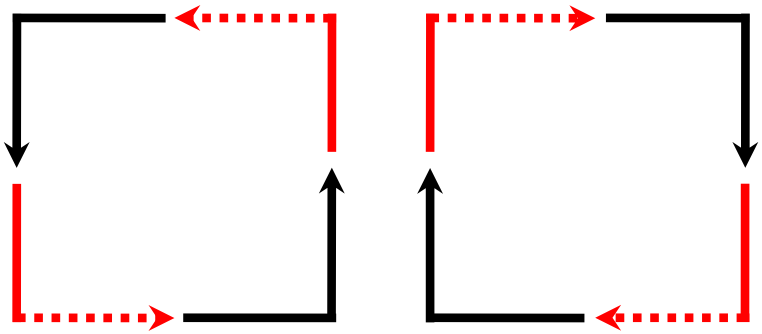
XY is deadlock free, YX is deadlock free, what about XY+YX?

No!

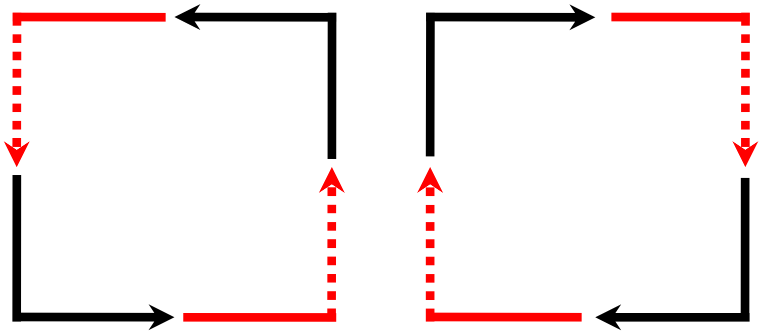
DOR – Turns allowed

- One way of looking at whether a routing algorithm is deadlock-free is to look at the turns allowed

- Deadlocks may occur if turns can form a cycle



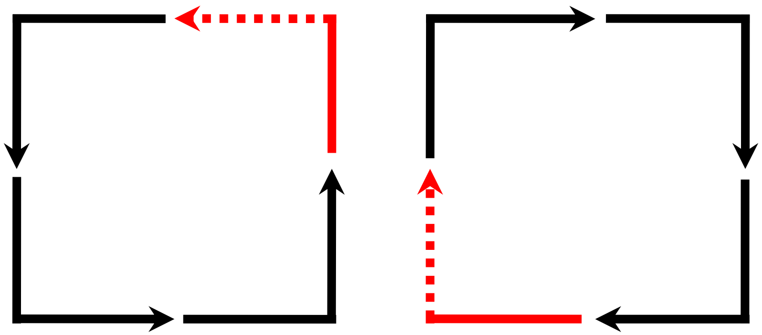
XY Model



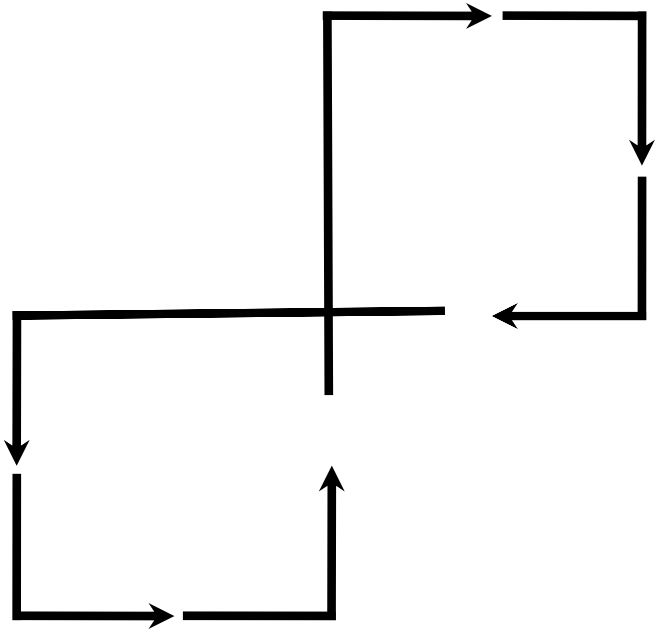
YX Model

Allowing more turns

- Allowing more turns may allow adaptive routing, but also deadlock



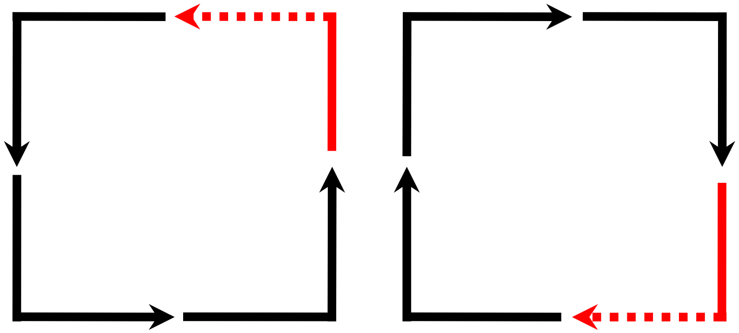
Six turn model



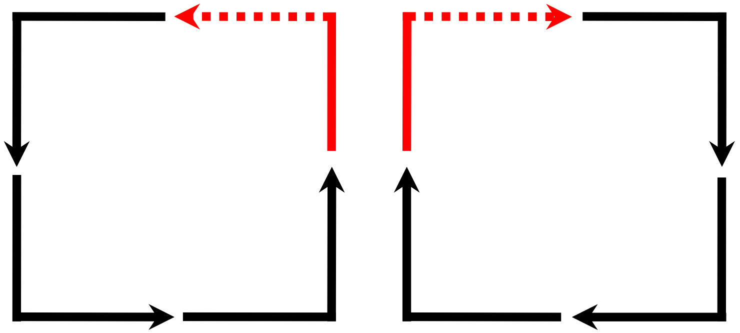
Turn Model [Glass and Ni, 1994]

- A systematic way of generating deadlock-free routes with small number of prohibited turns

- Deadlock-free if routes conform to at least ONE of the turn models (acyclic channel dependence graph)



West-First Turn Model

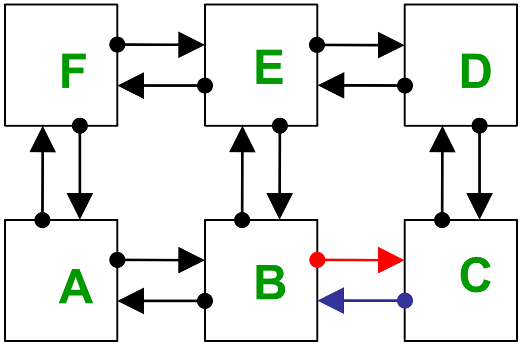


North-Last Turn Model

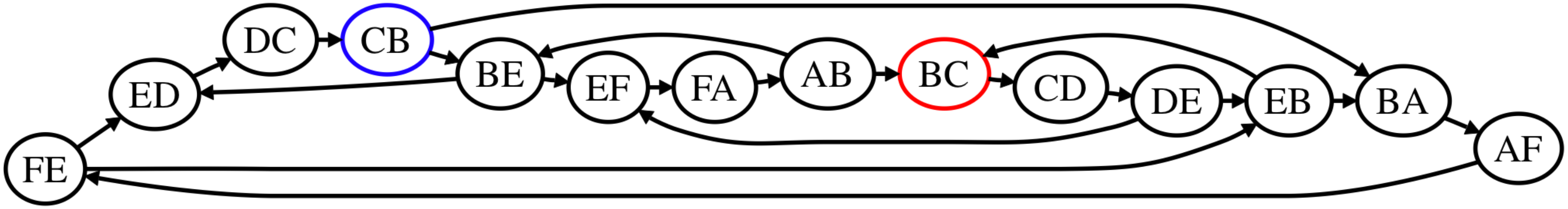
2-D Mesh and CDG

Can create a channel dependency graph (CDG) of the network.

Vertices in the CDG represent network *links*

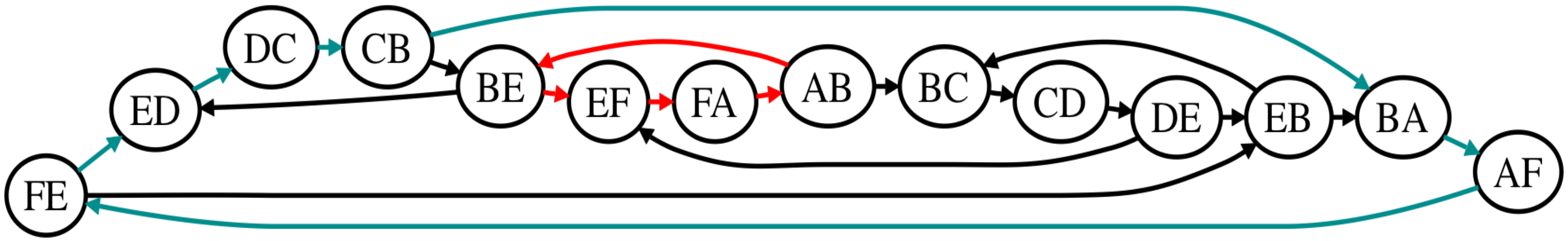


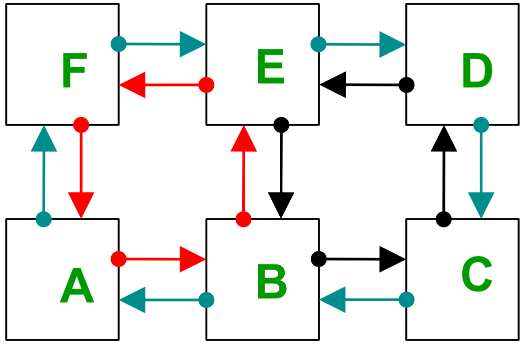
Disallowing 180° turns, e.g., AB → BA



Cycles in CDG

The channel dependency graph D derived from the network topology may contain many cycles

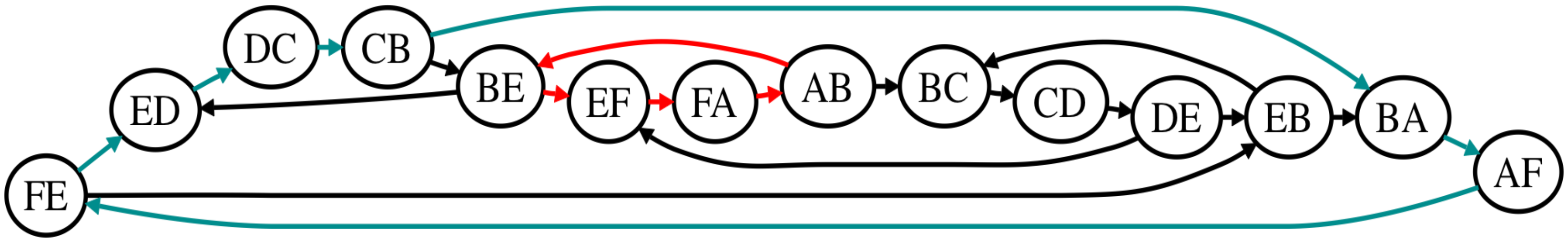


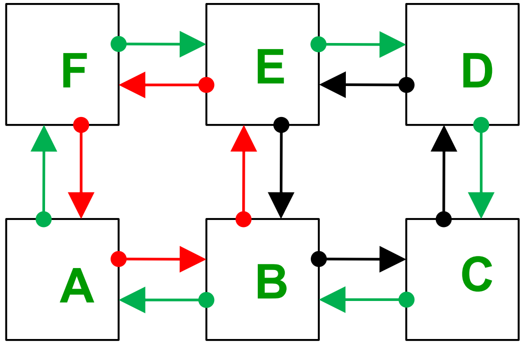


Flow routed through links AB, BE, EF
Flow routed through links EF, FA, AB
Deadlock!

Key Insight

If routes of flows conform to acyclic CDG, then there will be no possibility of deadlock!



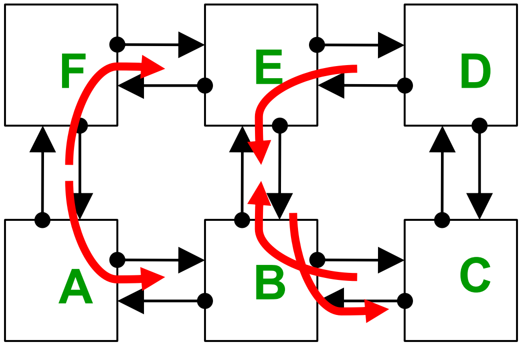


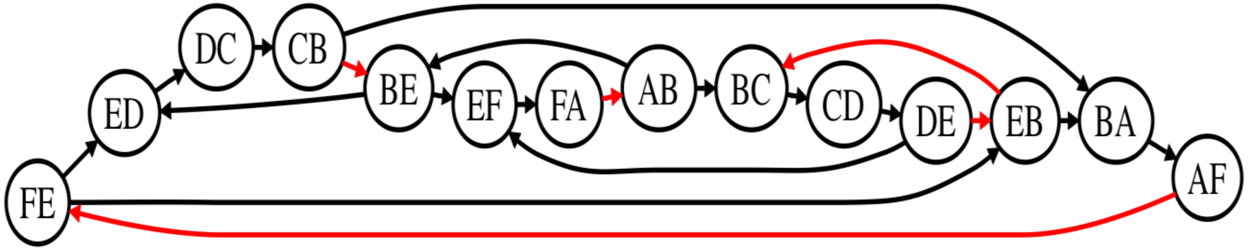
Disallow/Delete certain edges in CDG

Edges in CDG correspond to turns in network!

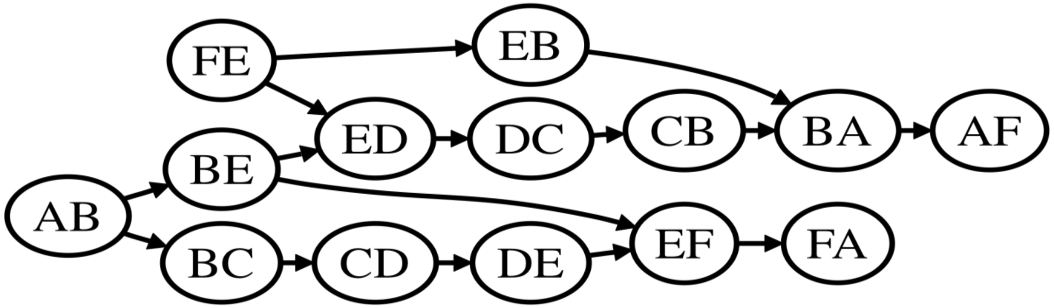
Acyclic CDG → Deadlock-free routes

Turns could be prohibited ad-hoc, all the edges in red are deleted



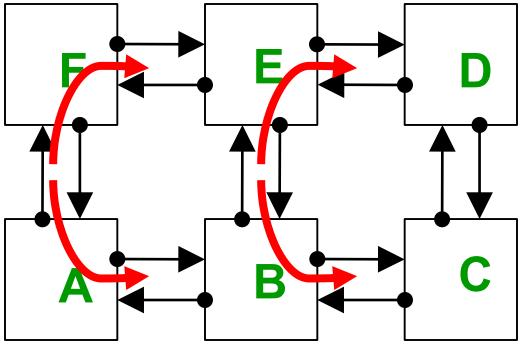


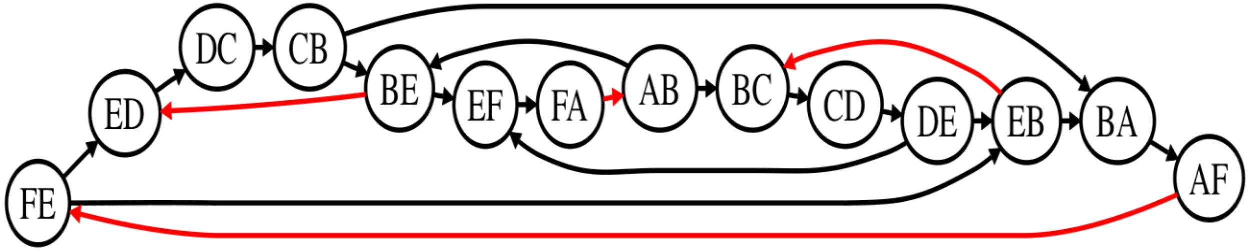
Ad-hoc Acyclic CDG



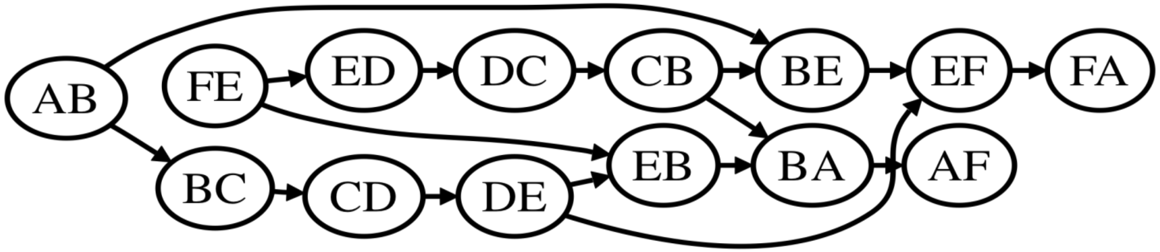
West-first → Deadlock-free routes

Per the West-First prohibited turns, all the edges in red are deleted

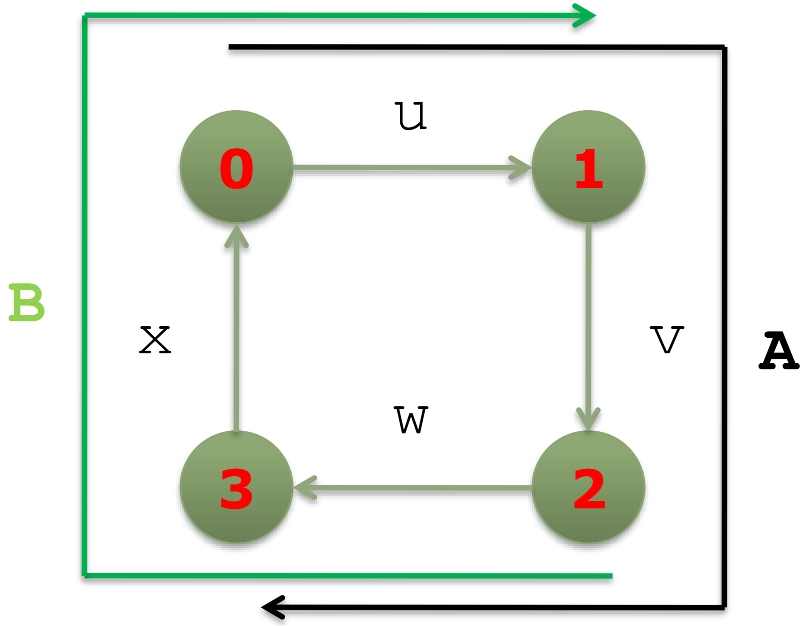




West-First Acyclic CDG



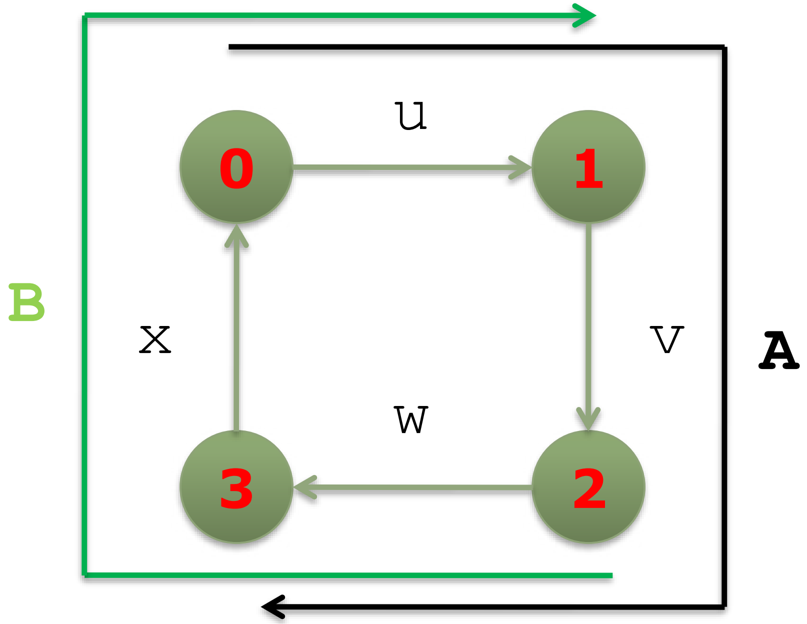
Resource Conflicts → Deadlock



Routing deadlocks in wormhole routing result from Structural hazard at router resources, e.g., buffers.

How can structural hazards be avoided?

Resource Conflicts → Deadlock



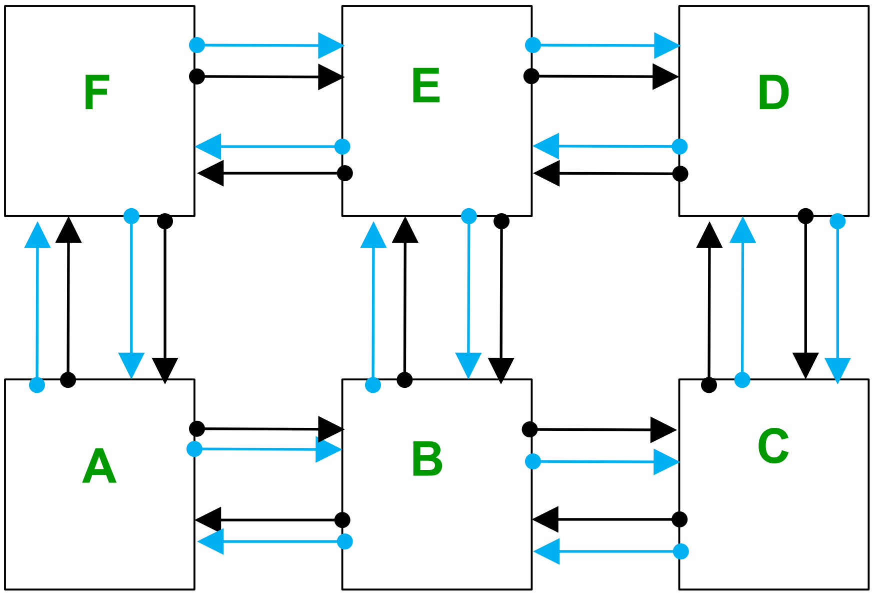
Routing deadlocks in wormhole routing result from Structural hazard at router resources, e.g., buffers.

How can structural hazards be avoided?

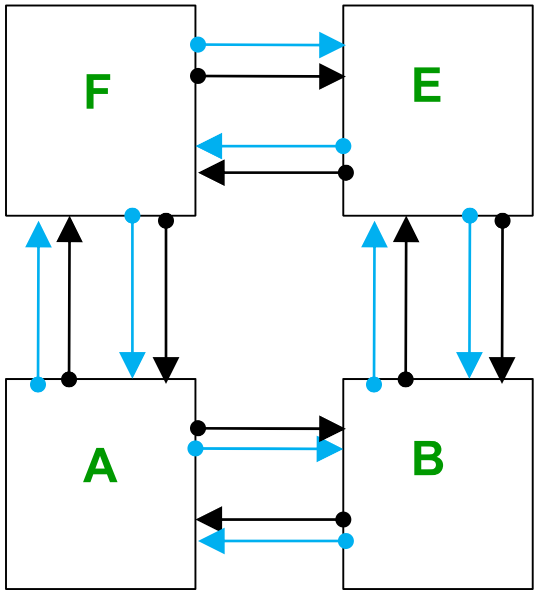
Adding more resources

Virtual Channels

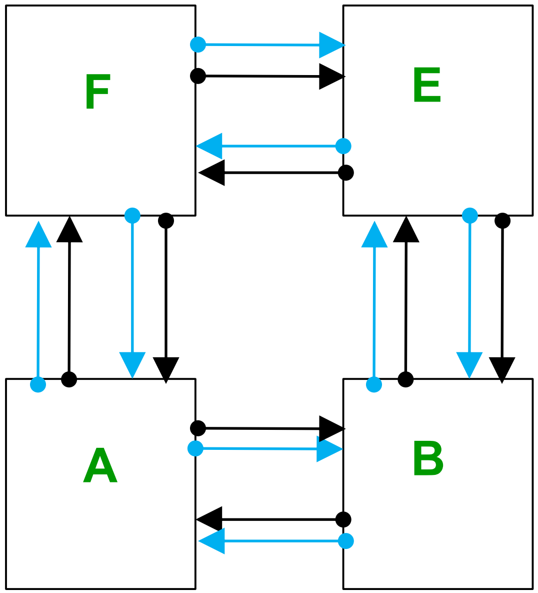
- Virtual channels can be used to avoid deadlock by restricting VC allocation

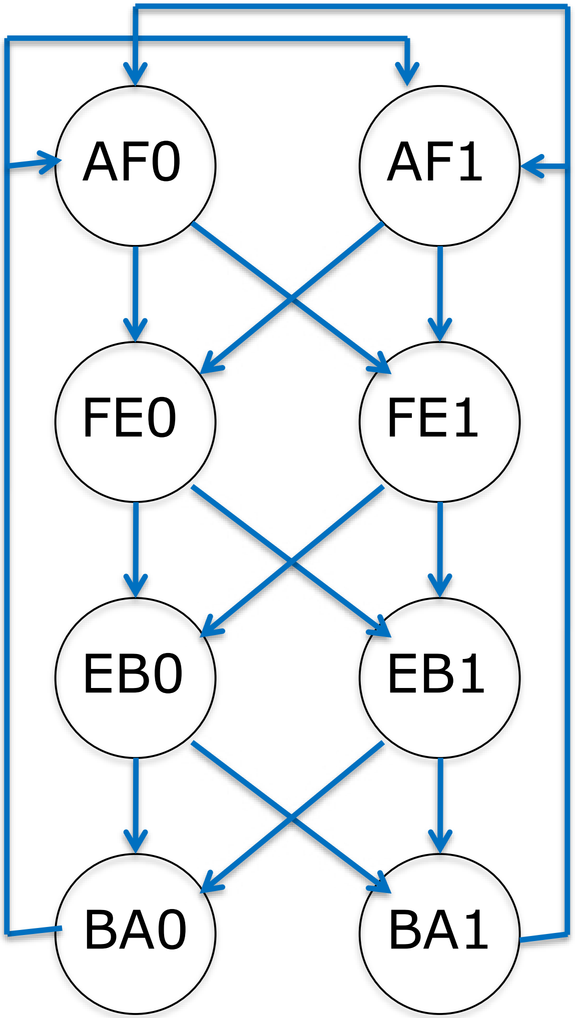


CDG and Virtual Channels

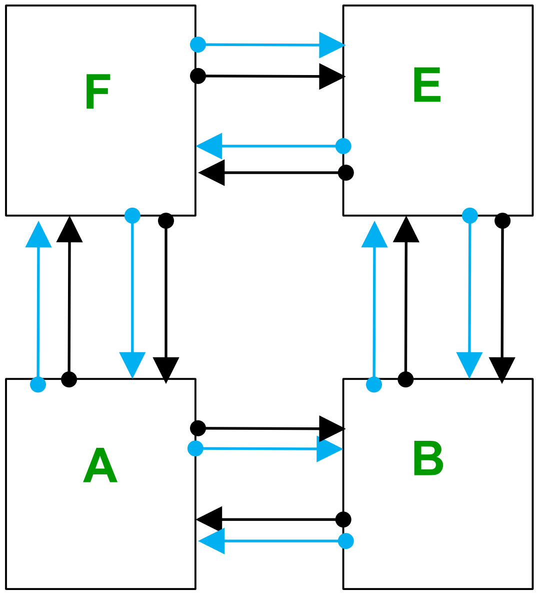


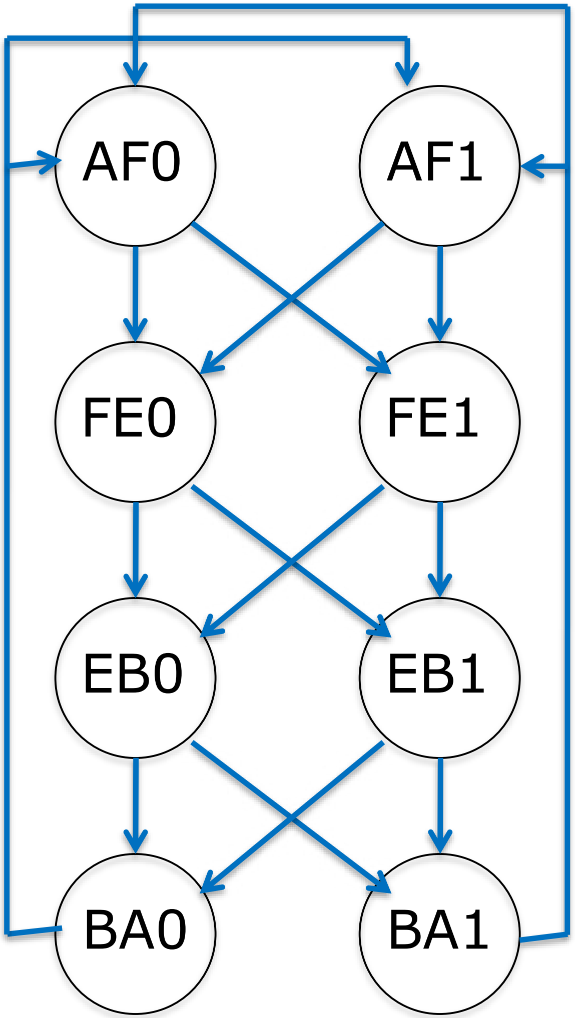
CDG and Virtual Channels

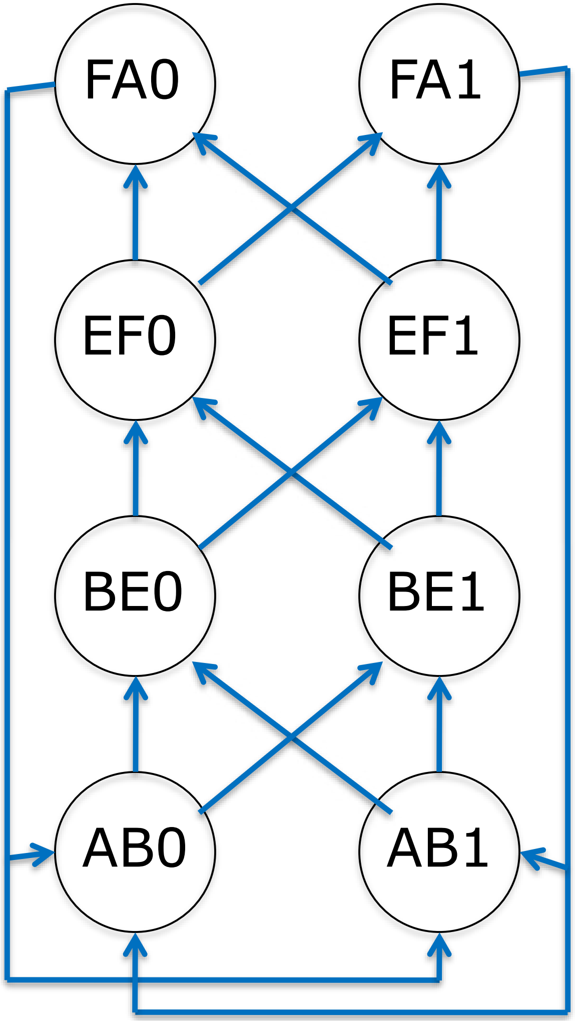




CDG and Virtual Channels

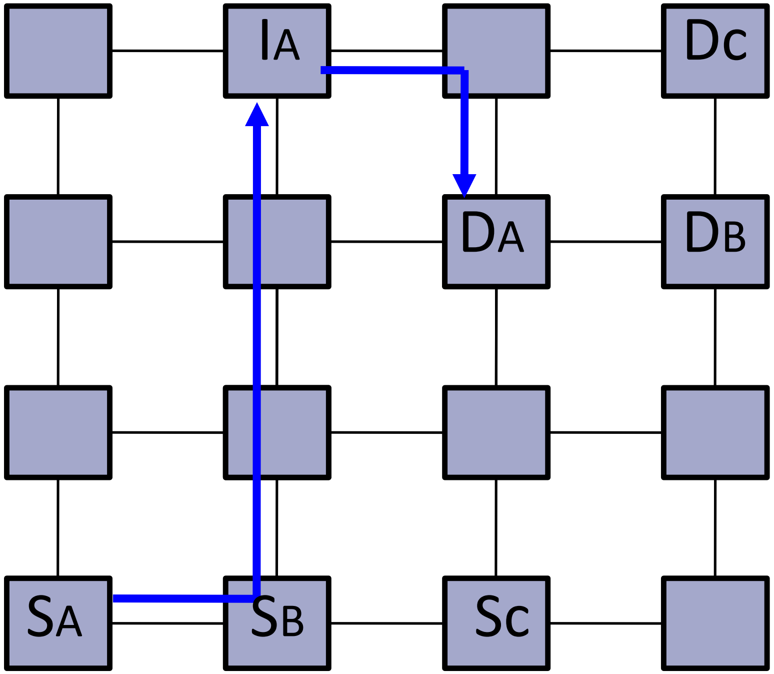






Randomized Routing: Valiant

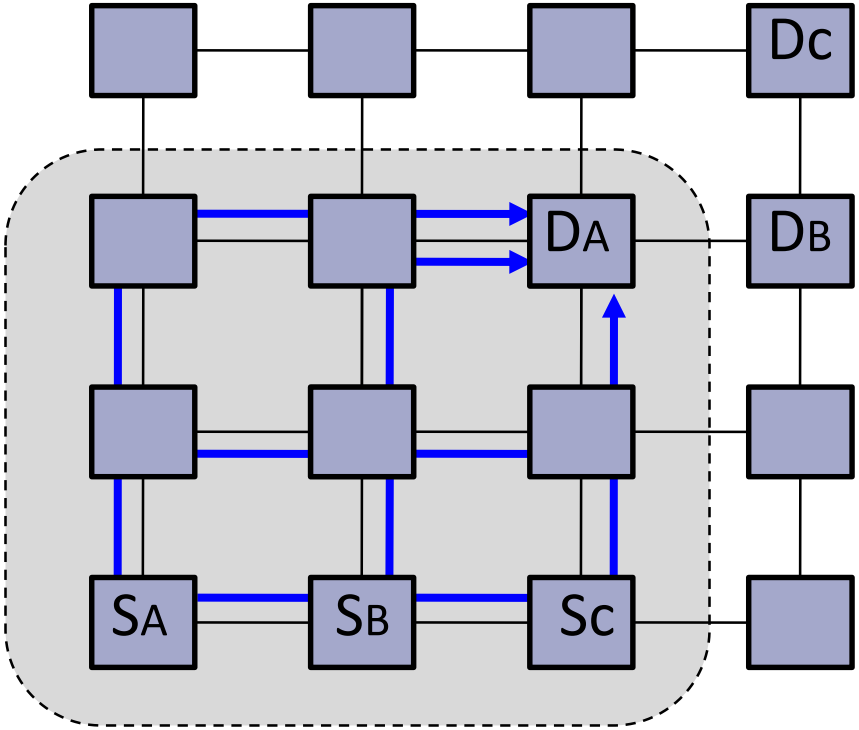
- Route each packet through a randomly chosen intermediate node



A packet, going from node SA to node DA, is first routed from SA to a randomly chosen intermediate node IA, before going from IA to final destination DA.

It helps load-balance the network and has a good worst-case performance at the expense of locality.

ROMM: Randomized, Oblivious Multi-phase Minimal Routing



To retain locality, choose intermediate node in the minimal quadrant

Equivalent to randomly selecting among the various minimal paths from source to destination

Thank you!

Next Lecture: VLIW