CS 223 Computer Organization & Architecture

Lecture 36 [15.05.2020]

NoC Router Microarchitecture

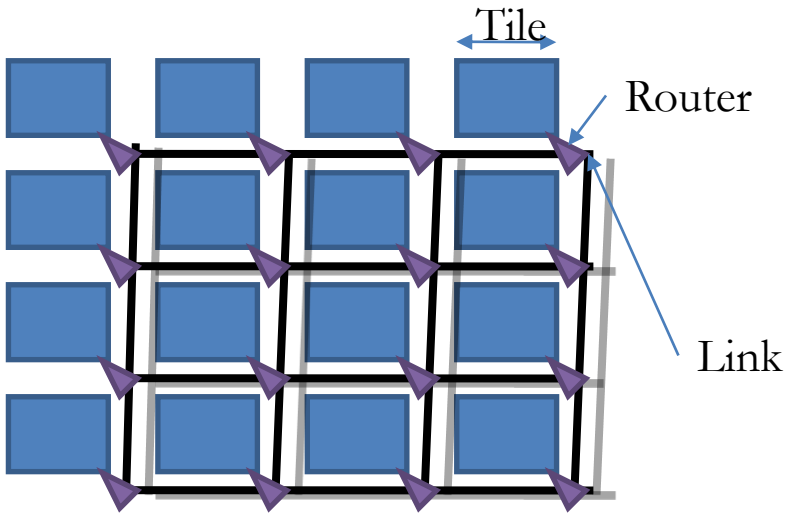
John Jose

Assistant Professor

Department of Computer Science & Engineering

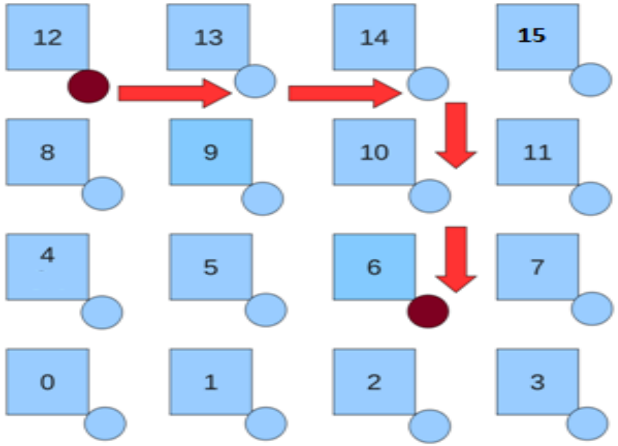
Indian Institute of Technology Guwahati, Assam.

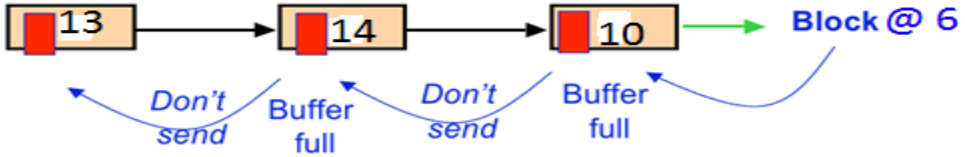
Building Blocks of NoC



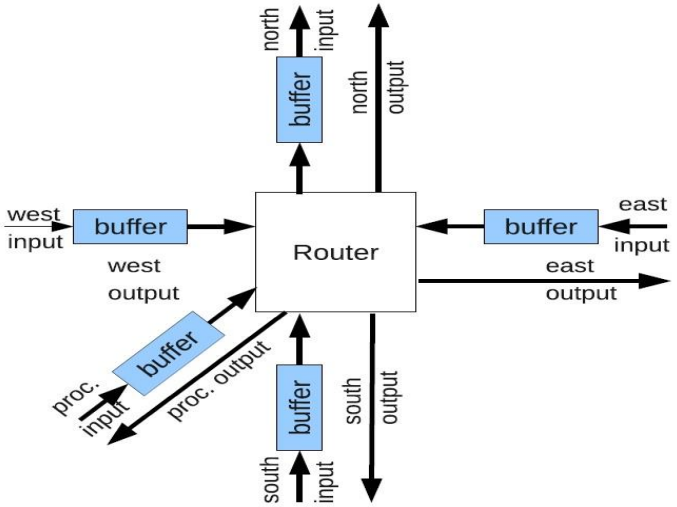
- Topology
- Routing
- Flow control
- Router micro-architecture

Flow Control





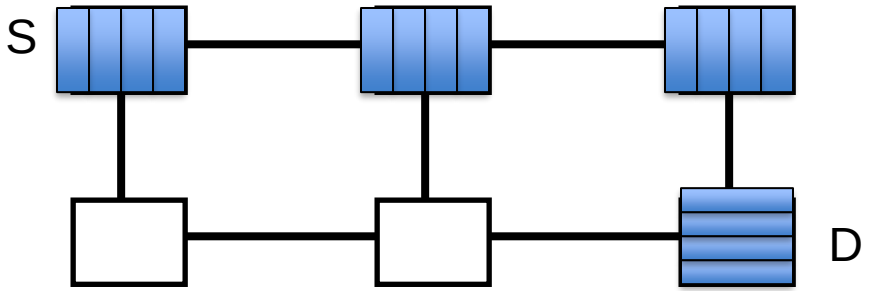
"Backpressure"



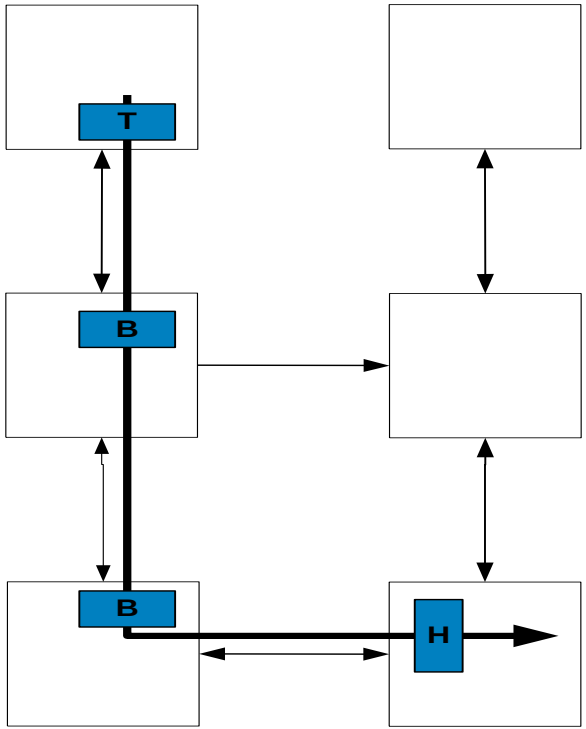
- Upstream router should know the buffer availability of downstream router.
- Credit should be exchanged between routers by handshake signals

Store and Forward Flow Control

- Store and Forward Packet based flow control
  - Packet copied entirely into network router before moving to the next node
  - Flow control unit is the entire packet
- Leads to high per-packet latency
- Requires buffering for entire packet in each node



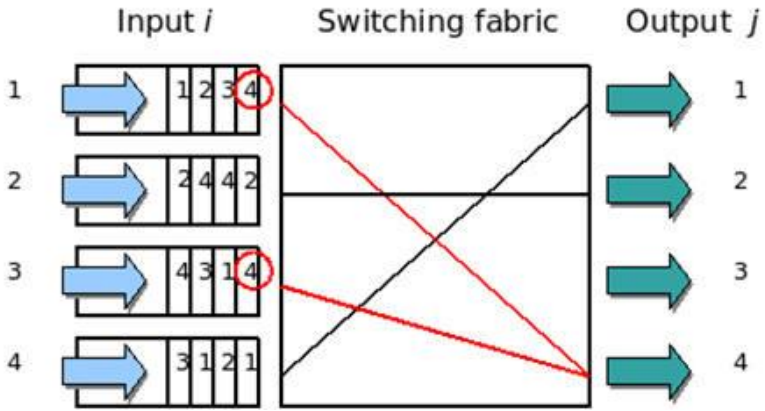
Wormhole Flow Control



- Packets broken into smaller flits
- Flits are sent across the fabric in a *wormhole fashion*
  - Body follows head, tail follows body
  - Pipelined
  - If head blocked, rest of packet stops
  - Routing (src/dest) information only in head
- Lower latency, efficient buffer utilization
- Occupies resources across multiple routers

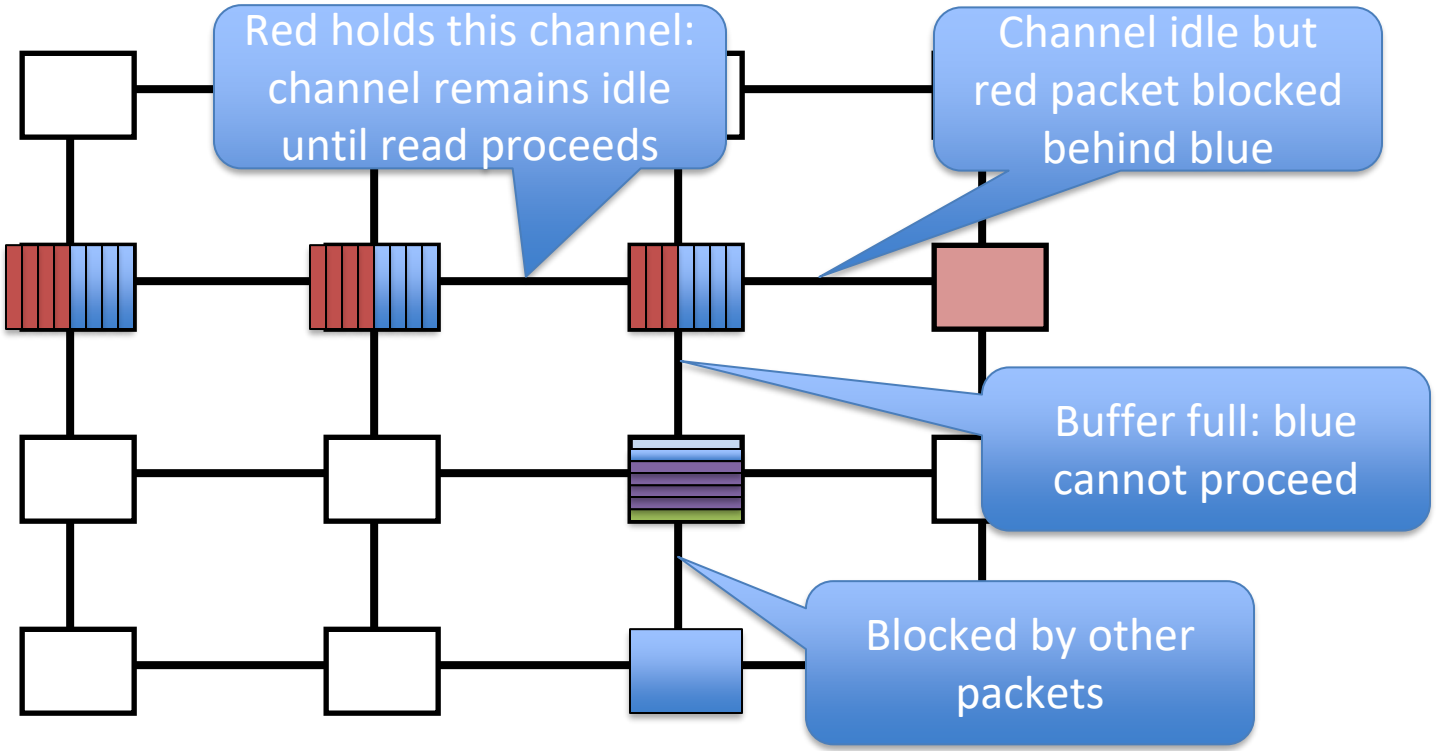
Head of Line Blocking

- Suffers from **head of line blocking**
  - If head flit cannot move due to contention, another worm cannot proceed even though links may be idle



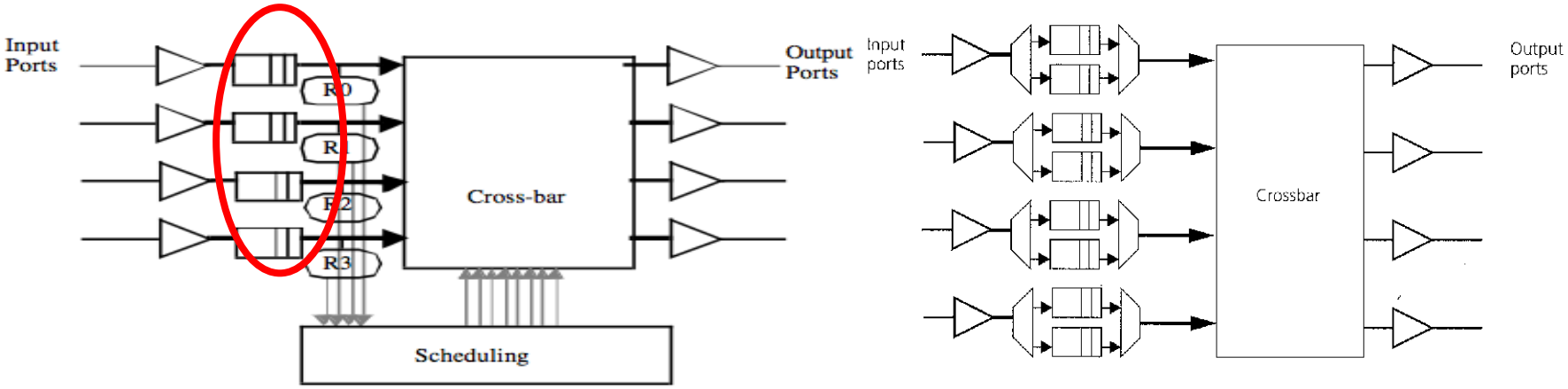
HOL Blocking

Head of Line Blocking

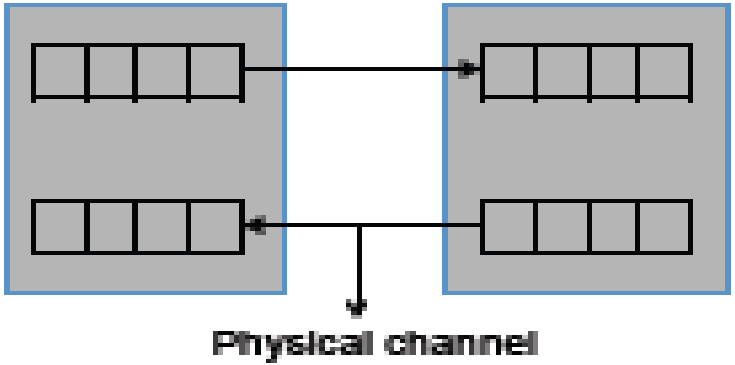


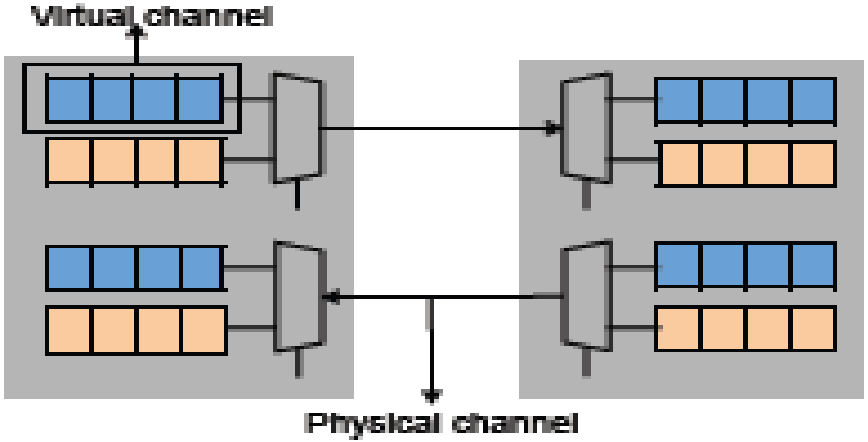
Virtual Channel Flow Control

- Multiplex multiple channels over one physical channel
- FIFO buffers replaced with multilane buffers
- Divide up the input buffer into multiple buffers sharing a single physical channel



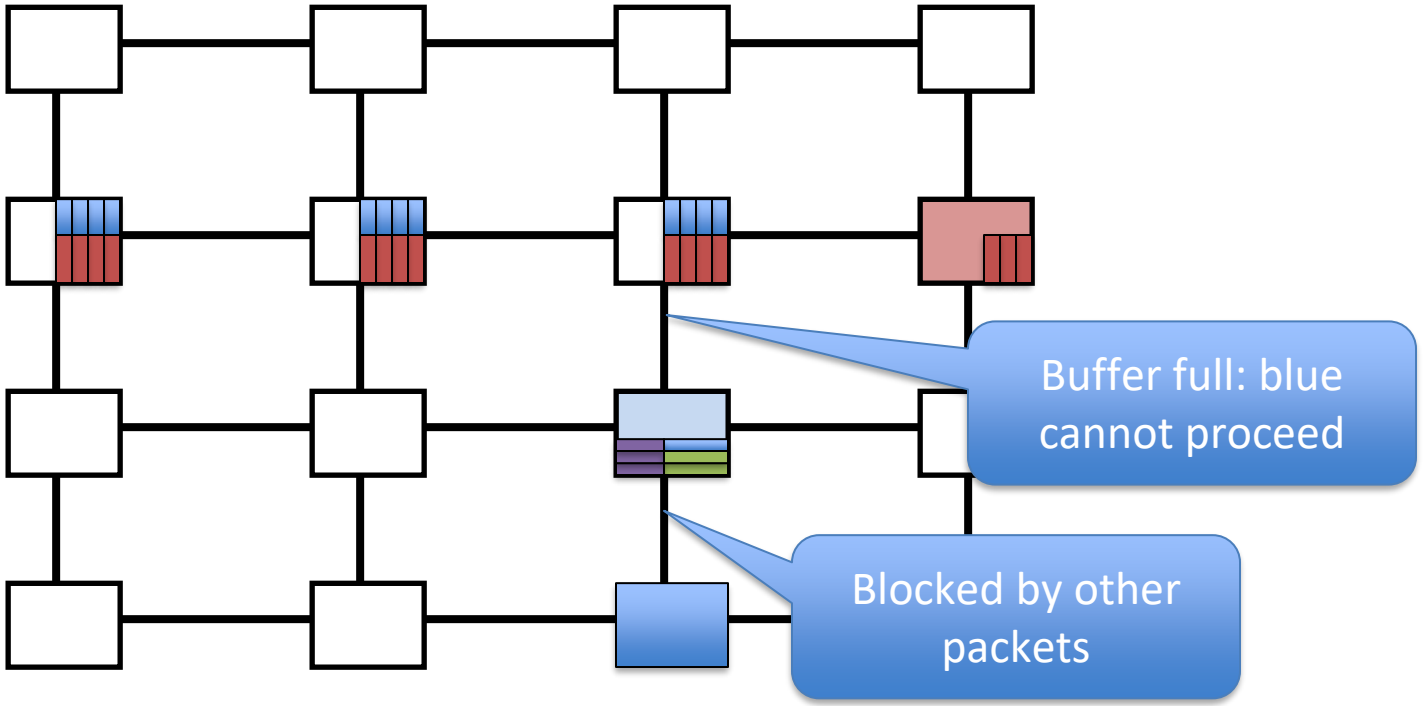
Virtual Channel Flow Control



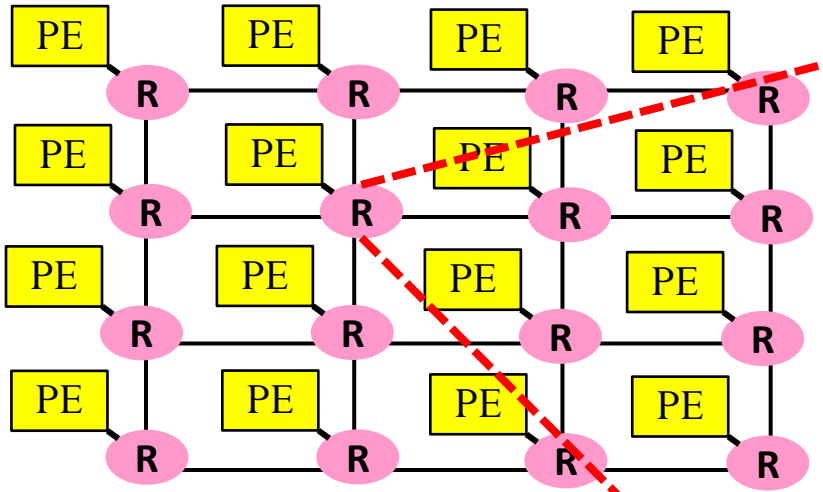


- VCs are allocated once at each router to the head flit and remaining flits of the packet inherit the same VC
- Flits of different packets can be interleaved on the same physical channel
- VCs avoid deadlocks

Virtual Channel Flow Control



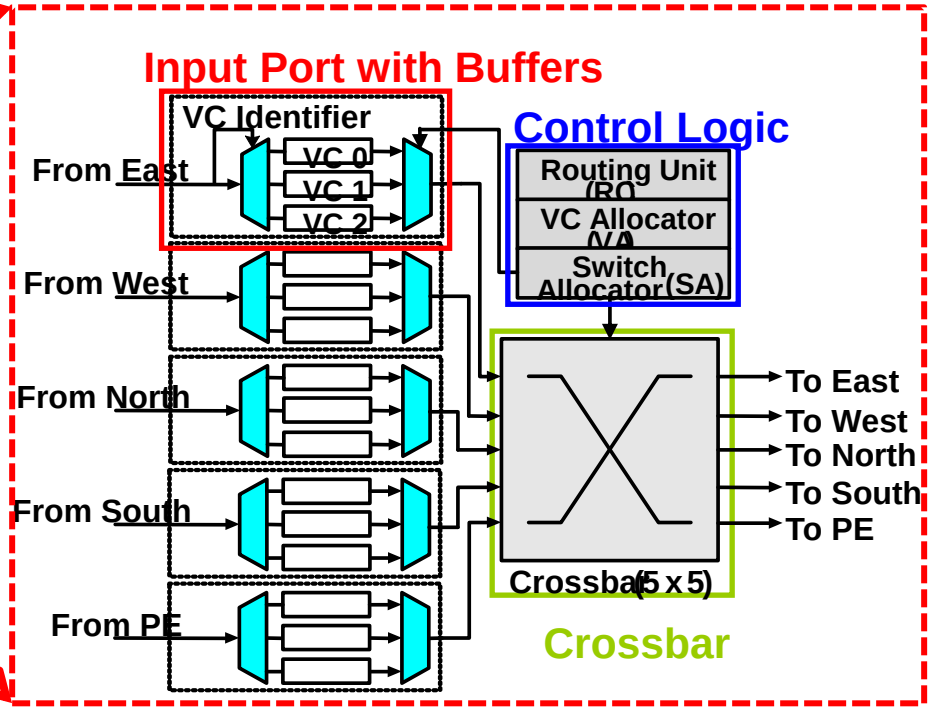
Input Buffered NoC Router



R Router

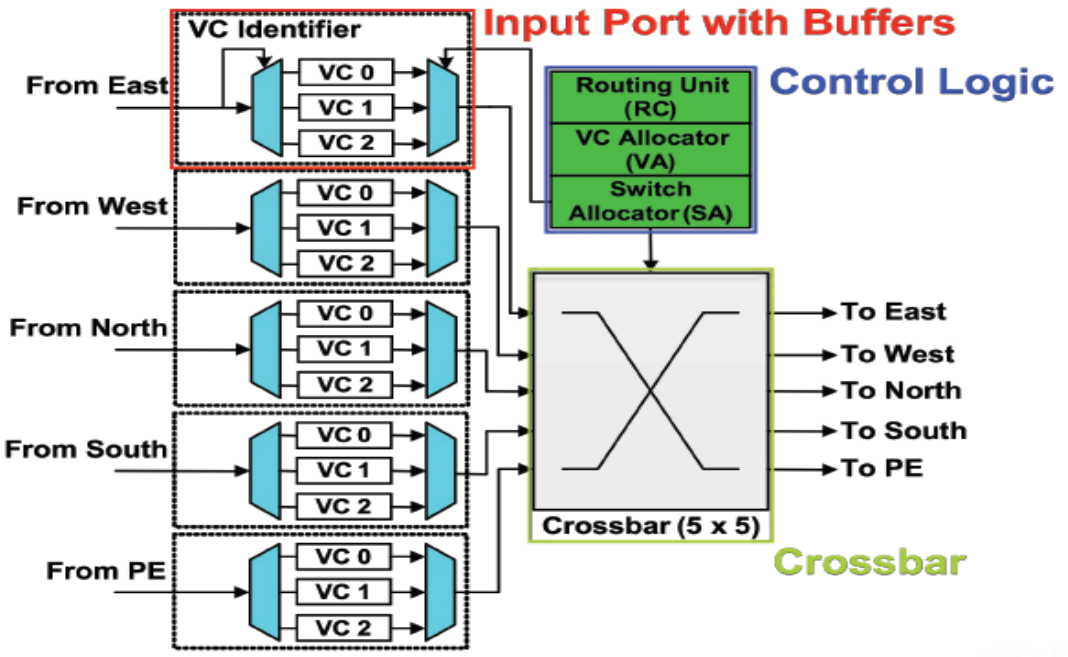
PE Processing Element

(Cores, L2 Banks, Memory Controllers etc)



Functions of a Router

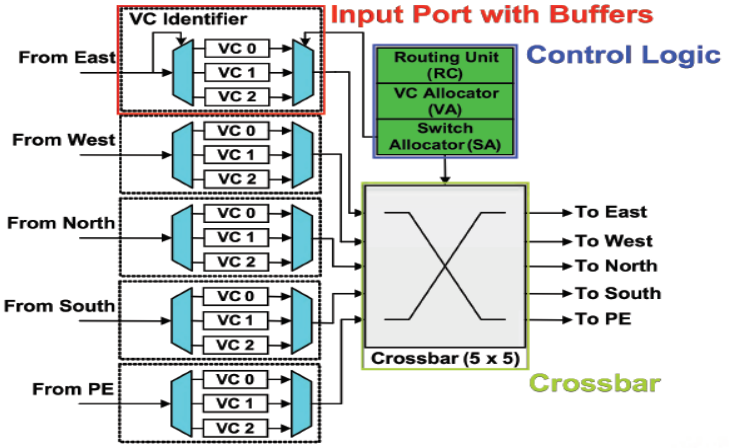
- Buffering of flits
- Route computation
- VC allocation
- Switch Allocation
- Switch Traversal



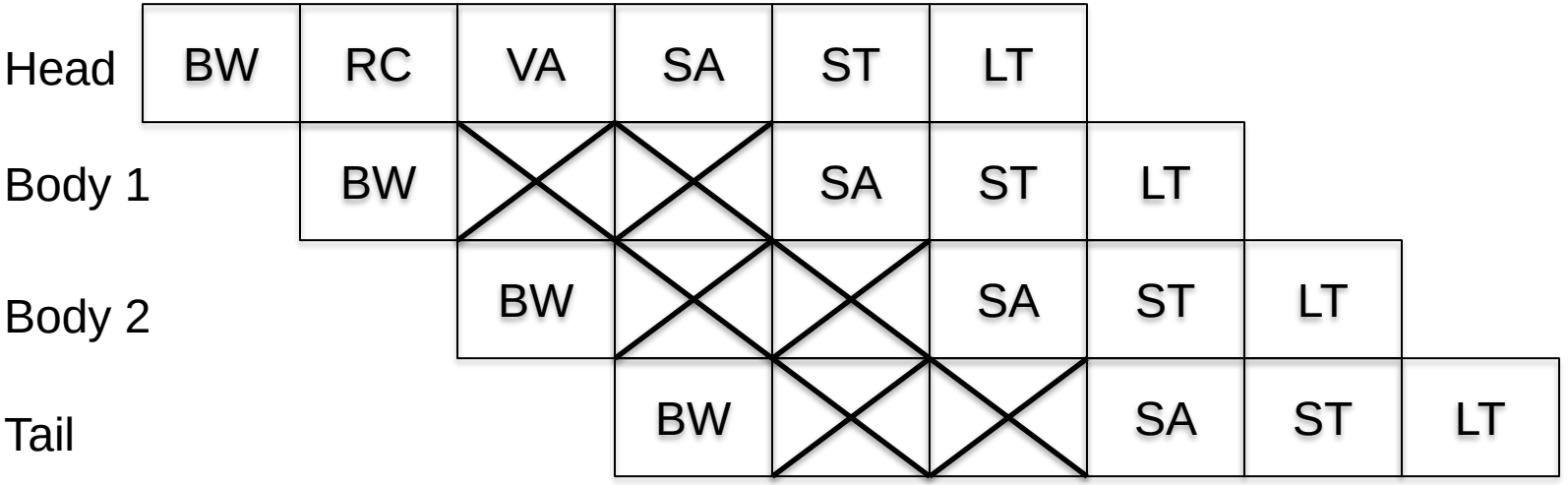
- Link Traversal

Router Pipeline

- Five logical stages
  - BW: Buffer Write
    - RC: Route computation
    - VA: Virtual Channel Allocation
    - SA: Switch Allocation
    - ST: Switch Traversal

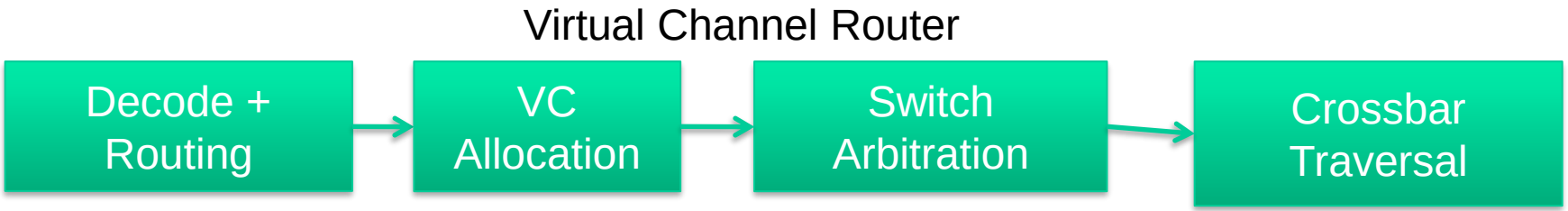


Wormhole Router Timeline



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

Dependencies in a Router



- Dependence between output of one module and input of another
  - Determine critical path through router
  - Cannot bid for switch port until routing performed

Lookahead Routing

- At current router perform routing computation for next router
  - Overlap with BW

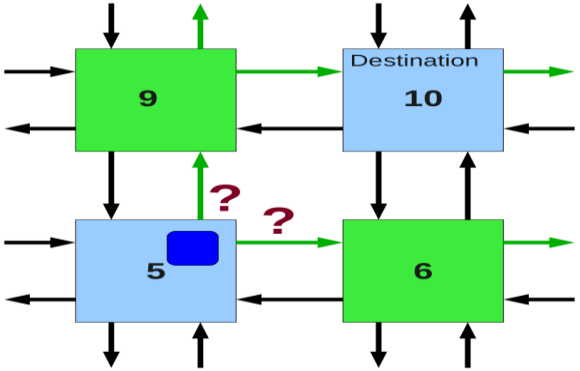
  - Pre-computing route allows flits to compete for VCs immediately after BW
  - RC decodes route header
  - Routing computation needed at next hop
    - Can be computed in parallel with VA

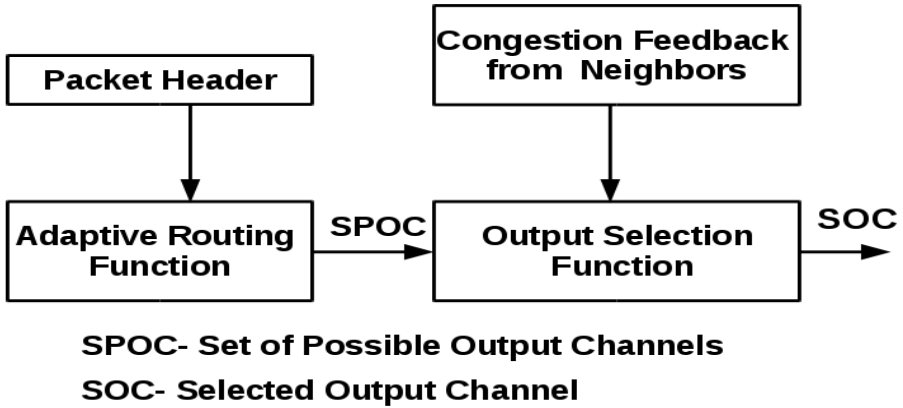
Speculative Routing

- Assume that Virtual Channel Allocation stage will be successful
  - Valid under low to moderate loads
- Entire VA and SA in parallel
- If VA unsuccessful (no virtual channel returned)
  - Must repeat VA/SA in next cycle

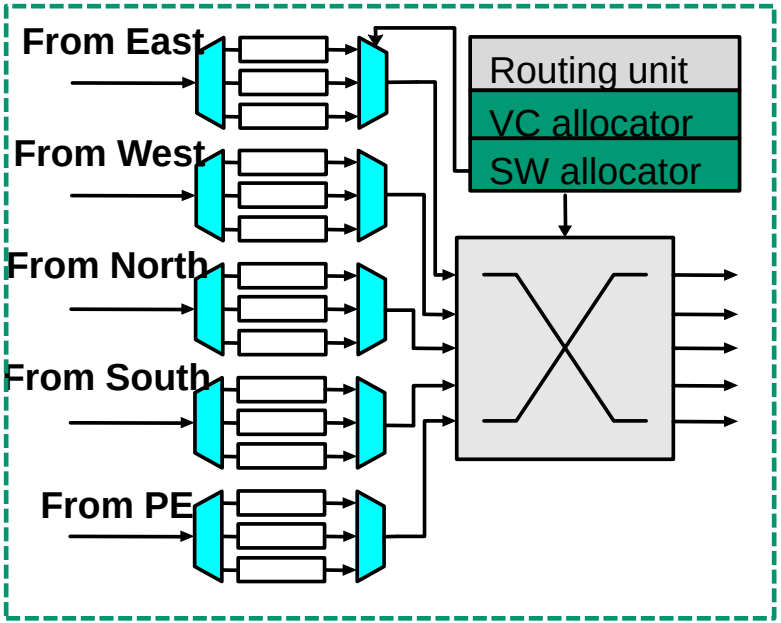
Selection Strategy

When there are multiple possible paths for a packet at a router, which one to choose ?

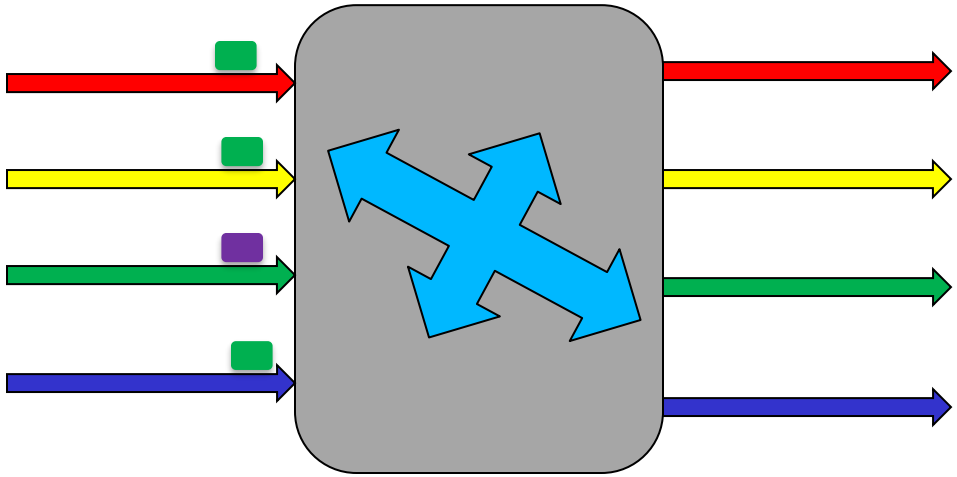




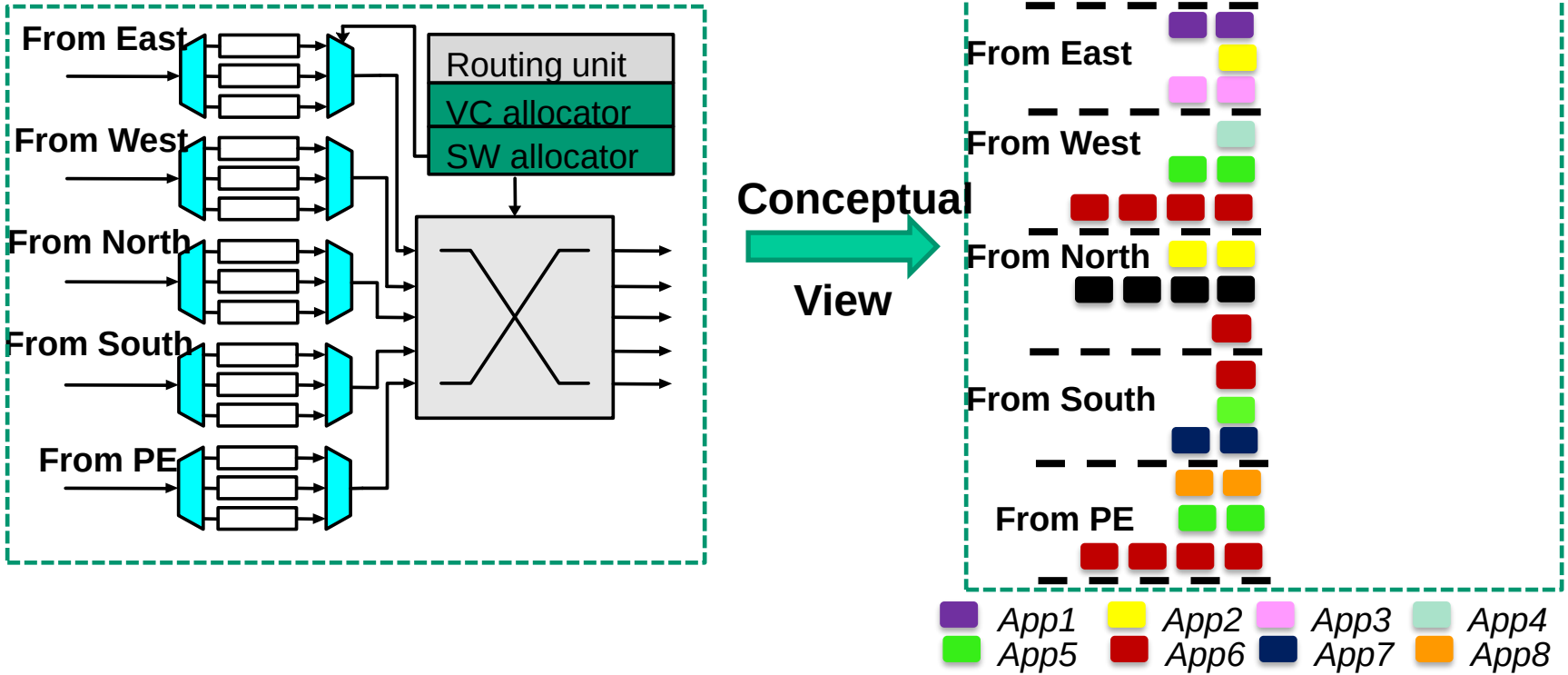
Input / Output Channel Selection



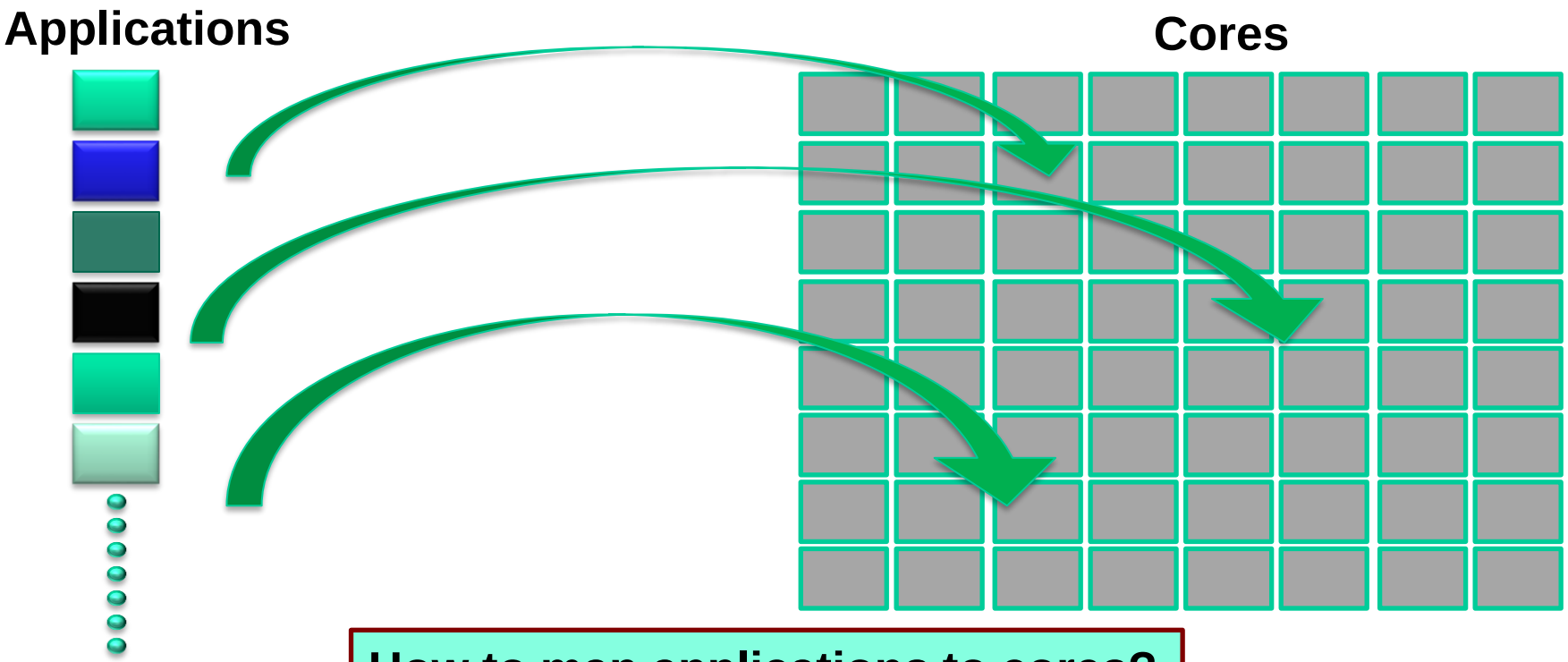
Output Channel Selection
Input Channel Selection



Switch Level Packet Scheduling in NoC



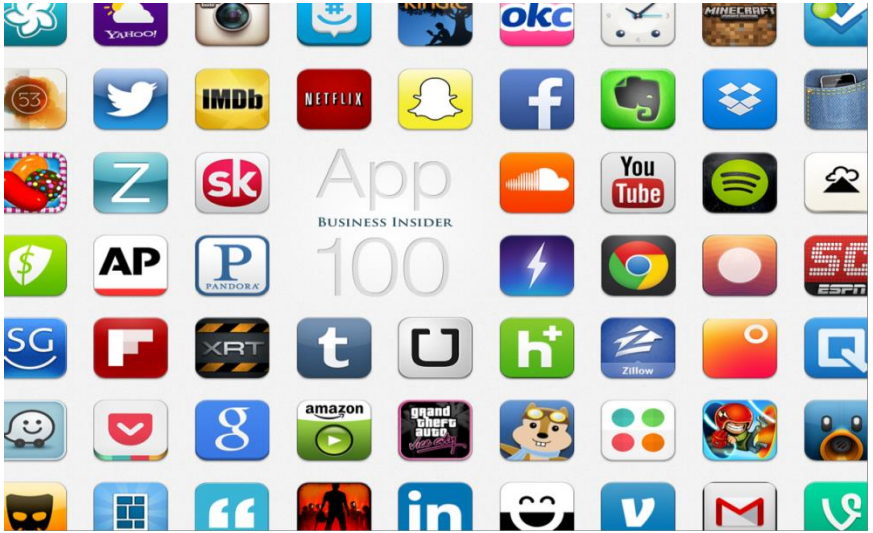
Application-to-Core Mapping Policies



How to map applications to cores?

Application-to-Core Mapping Policies

- **Application To Core Mapping**
  - **Clustering**
  - **Balancing**
  - **Isolation**
  - **Radial mapping**

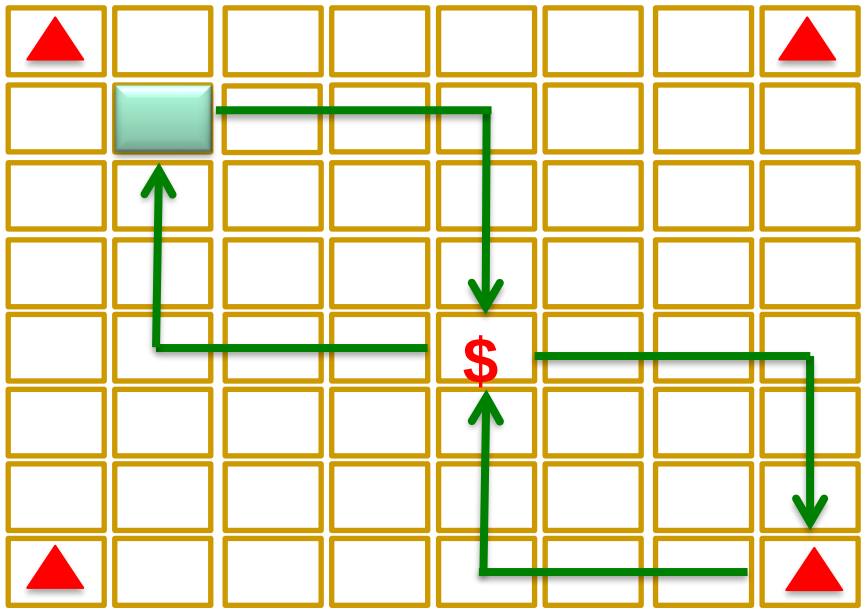


Task Scheduling

- Traditional
  - When to schedule a task? – Temporal
- Many-Core
  - When to schedule a task? – Temporal
  - Where to schedule a task? **– Spatial**
- Spatial scheduling impacts performance of memory hierarchy
- Latency is impacted by interference in NoC, memory, and caches

On-Chip Communication

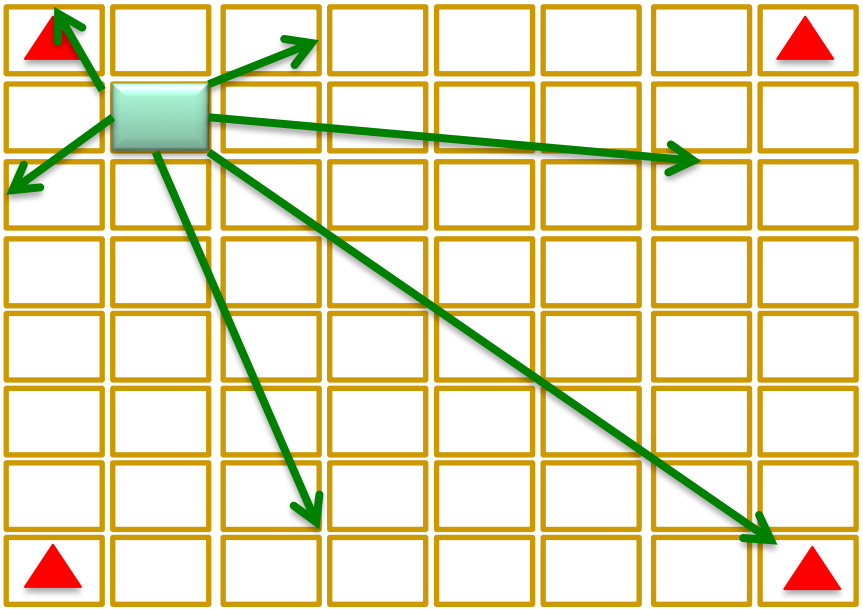
Application



Memory Controller

$ Shared Cache Bank

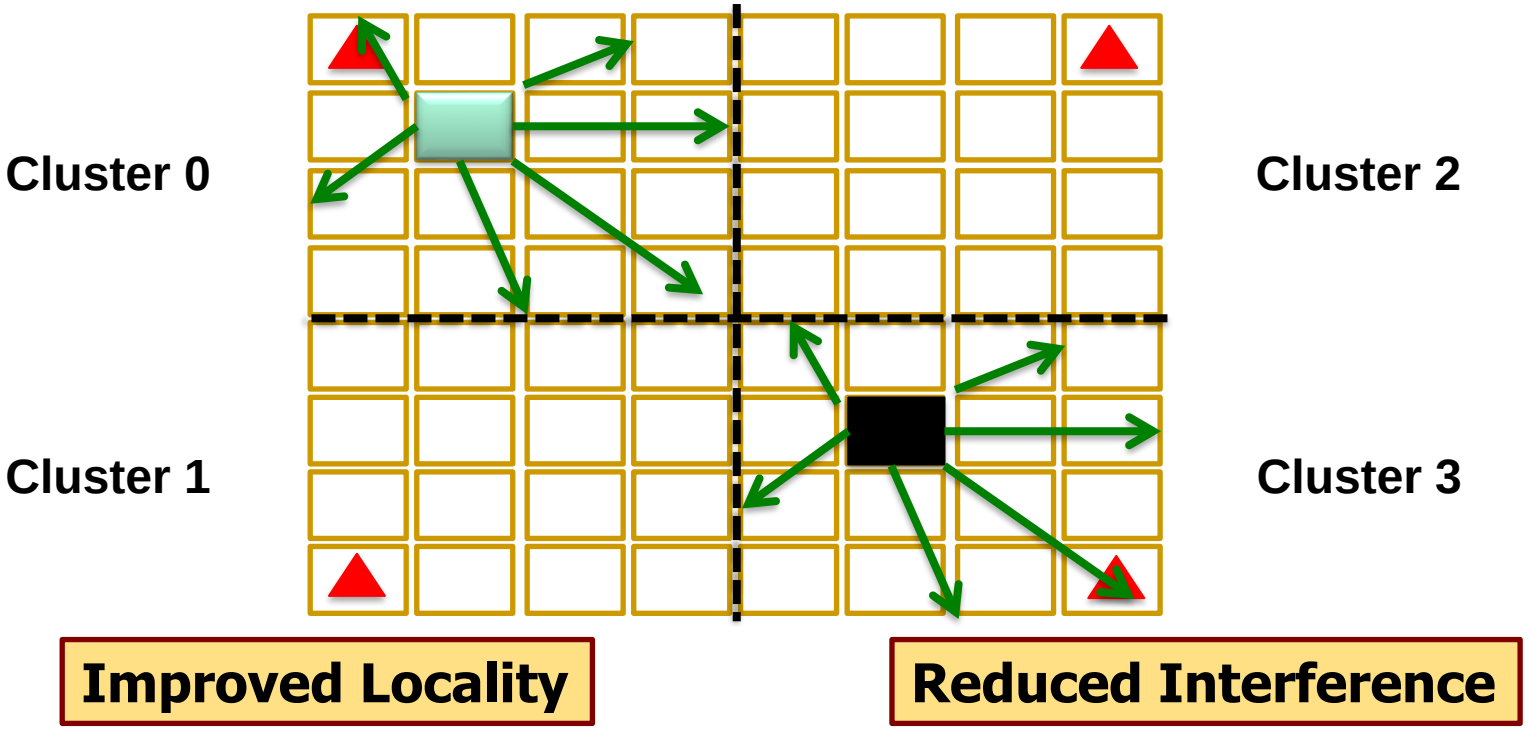
Clustering



Memory Controller

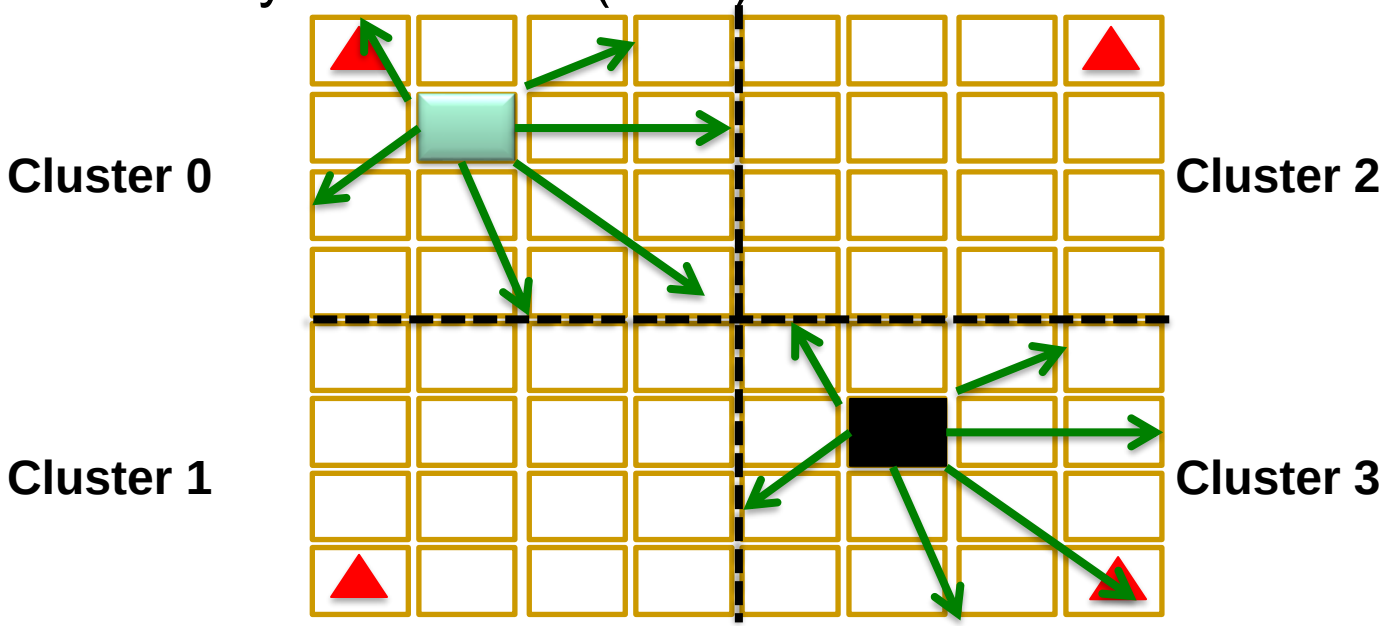
Inefficient data mapping to memory and caches

Clustering


Cluster 0
Cluster 2
Cluster 1
Cluster 3
Improved Locality
Reduced Interference

Clustering

- Locality aware page replacement policy
- When allocating free page, give preference to pages belonging to the cluster's memory controllers (MCs)



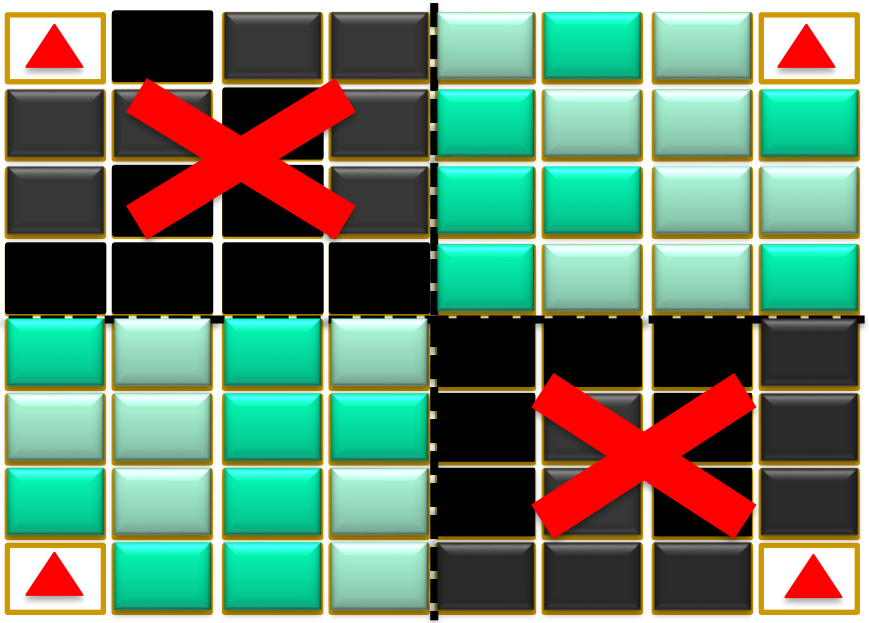
Balancing

Applications

Heavy

Light

Cores



Too much load in clusters with heavy applications

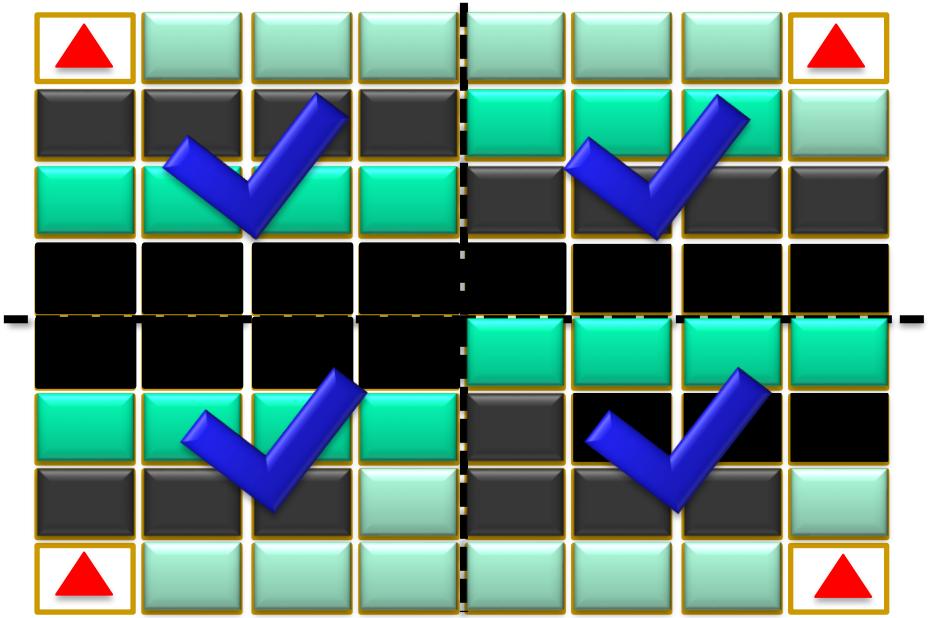
Balancing

Applications

Heavy

Light

Cores



Isolation

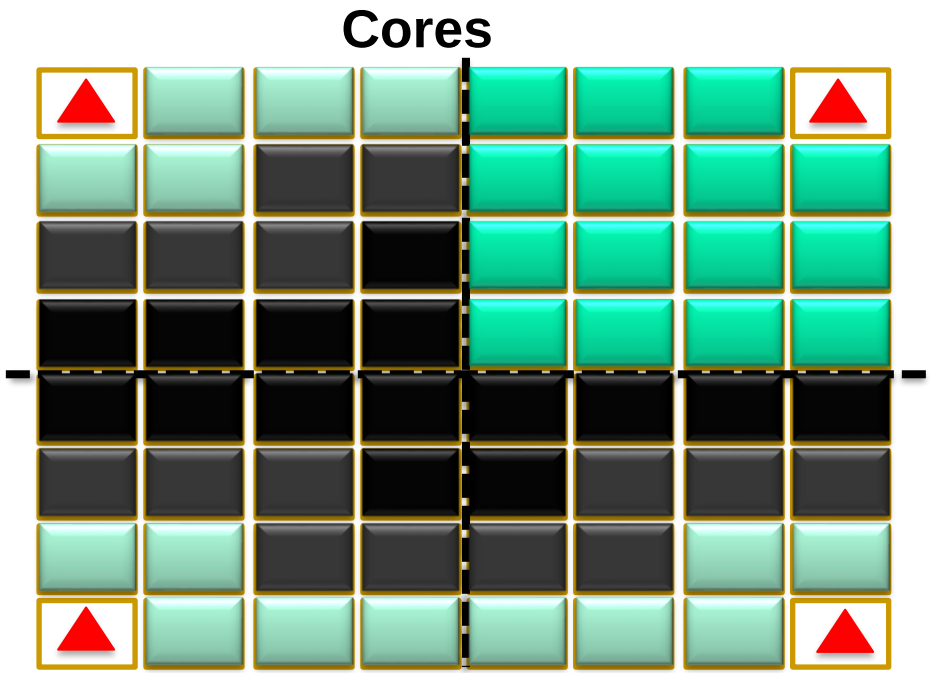
Applications

Sensitive

Light

Medium

Heavy



Isolate sensitive applications to a cluster

Balance load for remaining applications across clusters

Isolation

- How to estimate sensitivity?
  - High Miss— high misses per kilo instruction (MPKI)
  - Low MLP— high relative stall cycles per miss (STPM)
  - Sensitive if MPKI > Threshold & relative STPM is high
- Whether to or not to allocate cluster to sensitive applications?

Radial Mapping

Applications

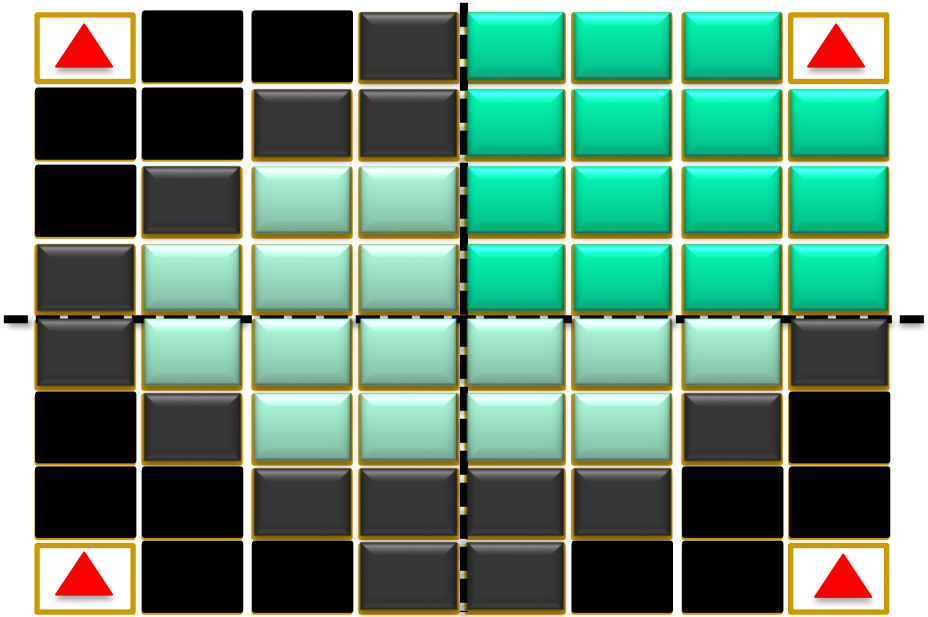
Sensitive

Light

Medium

Heav

Cores



Map applications that benefit most from being close to memory controllers close to these resources

johnjose@iitg.ac.in
http://www.iitg.ac.in/johnjose/