Lecture 14: Large Cache Design III

- Topics: Replacement policies, associativity, cache networks, networking basics

LIN

Qureshi et al., ISCA'06

- Memory level parallelism (MLP): number of misses that simultaneously access memory; high MLP → miss is less expensive
- Replacement decision is a linear combination of recency and MLP experienced when fetching that block
- MLP is estimated by tracking the number of outstanding requests in the MSHR while waiting in the MSHR
- Can also use set dueling to decide between LRU and LIN

Scavenger

Basu et al., MICRO'07

- Half the cache is used as a victim cache to retain blocks that will likely be used in the distant future
- Counting bloom filters to track a block's potential for reuse and make replacement decisions in the victim cache
- Complex indexing and search in the victim cache
- Another paper (NuCache, HPCA'11) places blocks in a large FIFO victim file if they were fetched by delinquent PCs and the block has a short re-use distance

V-Way Cache

Qureshi et al., ISCA'05

- Meant to reduce load imbalance among sets and compute a better global replacement decision
- Tag store: every set has twice as many ways
- Data store: no correspondence with tag store; need forward and reverse pointers
- In most cases, can replace any block; every block has a 2b saturating counter that is incremented on every access; scan blocks (and decrement) until a zero counter is found; continue scan on next replacement

ZCache

Sanchez and Kozyrakis, MICRO'10

- Skewed associative cache: each way has a different indexing function (in essence, W direct-mapped caches)
- When block A is brought in, it could replace one of four (say) blocks B, C, D, E; but B could be made to reside in one of three other locations (currently occupied by F, G, H); and F could be moved to one of three other locations
- We thus get a tree of replacement options and we can pick LRU among these options
- Every replacement requires multiple tag look-ups and data block copies; worthwhile if you're reducing off-chip accesses

Dead Block Prediction

- Can keep track of the number of accesses to a line during its previous residence; the block is deemed to be dead after that many accesses Kharbutli, Solihin, IEEE TOC'08
- To reduce noise, an access can be considered as a block's move to the MRU position Liu et al., MICRO 2008
- Earlier DBPs used a trace of PCs to capture when a block has completed its use
- DBP is used for energy savings, replacement policies, and cache bypassing

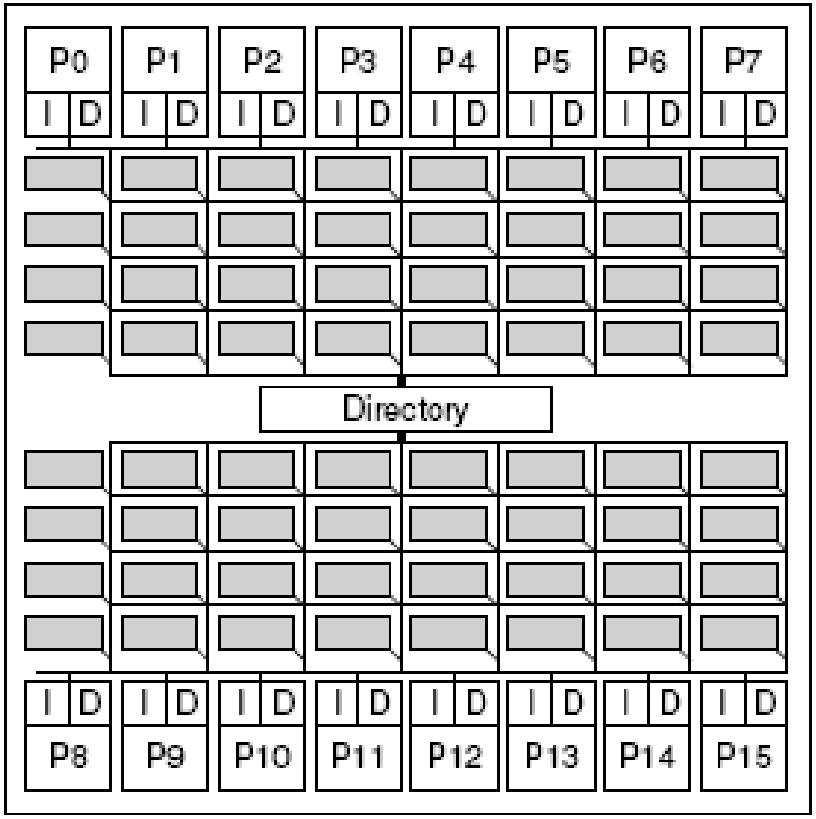
Distill Cache

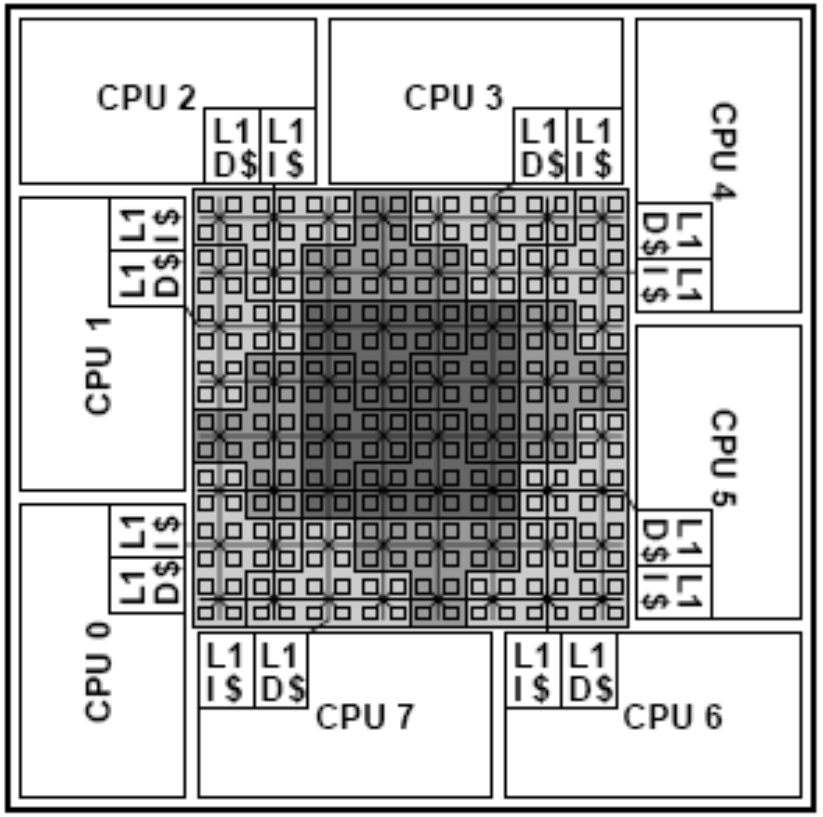
Qureshi, HPCA 2007

- Half the ways are traditional (LOC); when a block is evicted from the LOC, only the touched words are stored in a word-organized cache that has many narrow ways
- Incurs a fair bit of complexity (more tags for the WOC, collection of word touches in L1s, blocks with holes, etc.)
- Does not need a predictor; actions are based on the block's behavior during current residence
- Useless word identification is orthogonal to cache compression

Traditional Networks

Huh et al. ICS'05, Beckmann MICRO'04

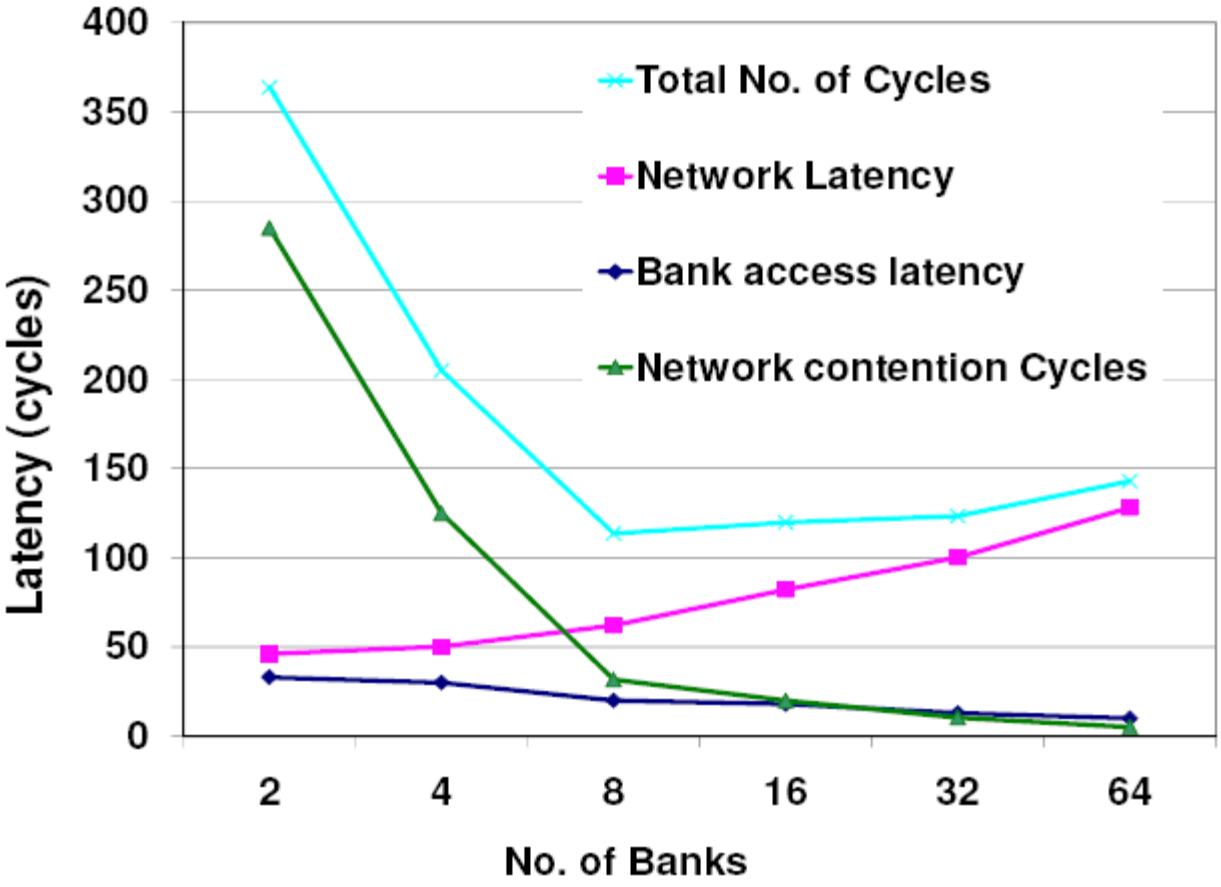


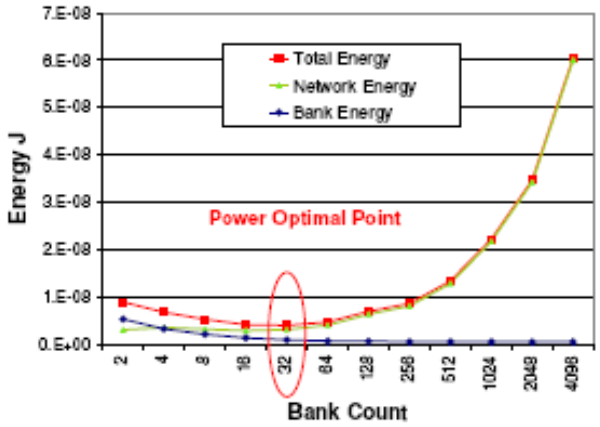


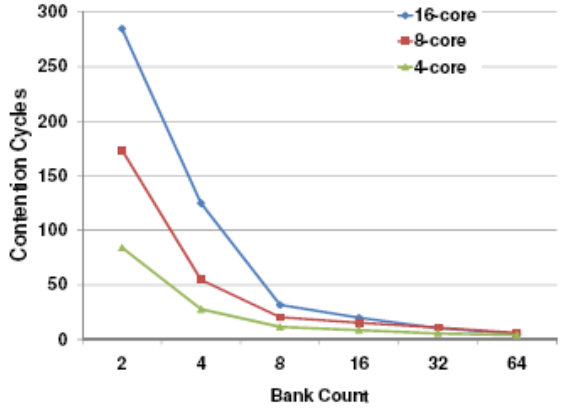
Example designs for contiguous L2 cache regions

Explorations for Optimality

Muralimanohar et al., ISCA'07

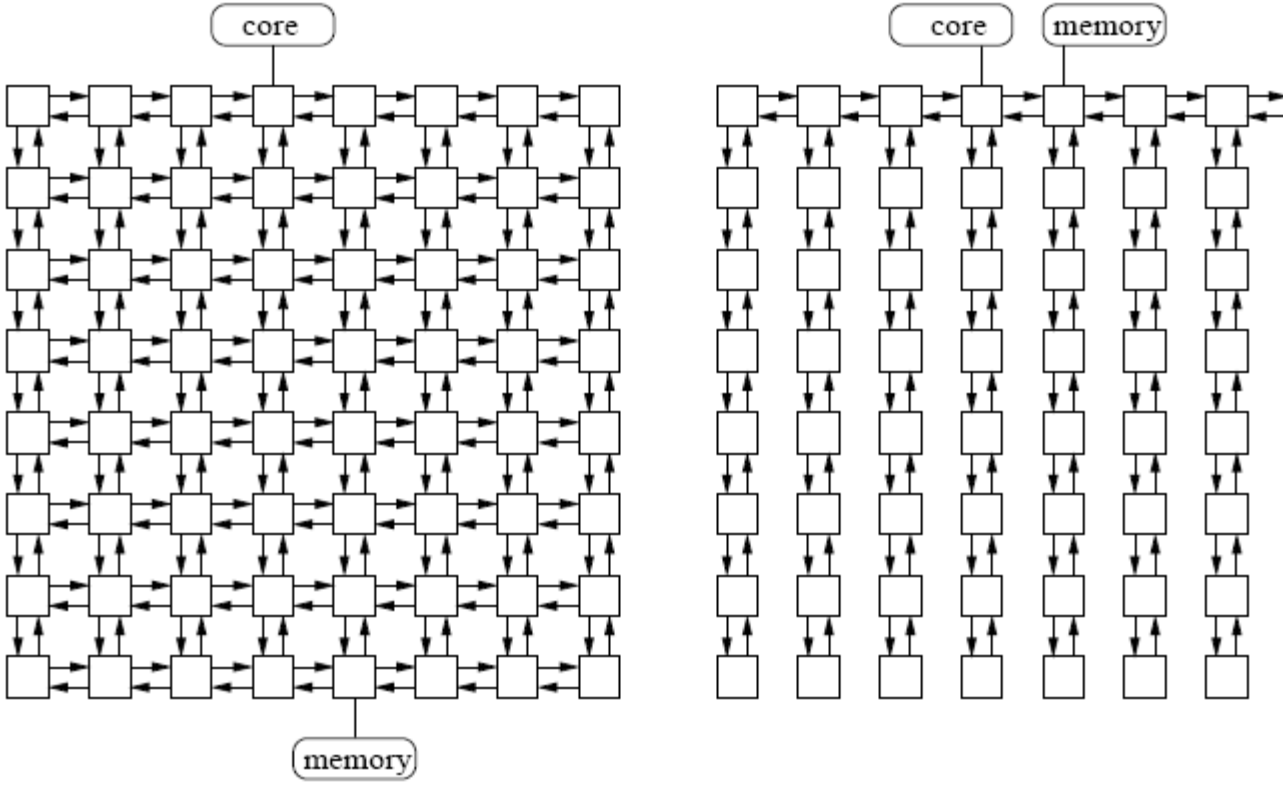






Halo Network

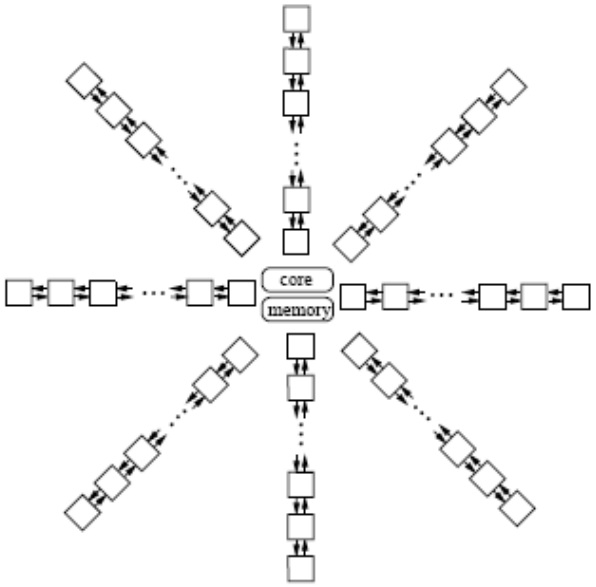
Jin et al., HPCA'07



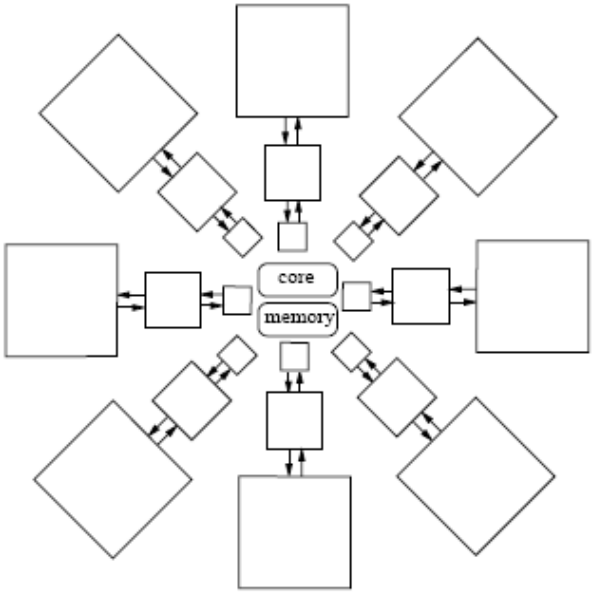
(a) Mesh (b) Simplified Mesh

- D-NUCA: Sets are distributed across columns;
  Ways are distributed across rows

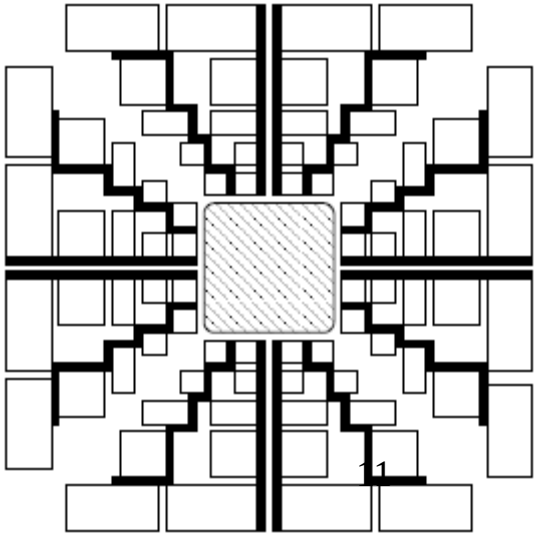
Halo Network



(c) Halo Constructed with Uniform Size Banks



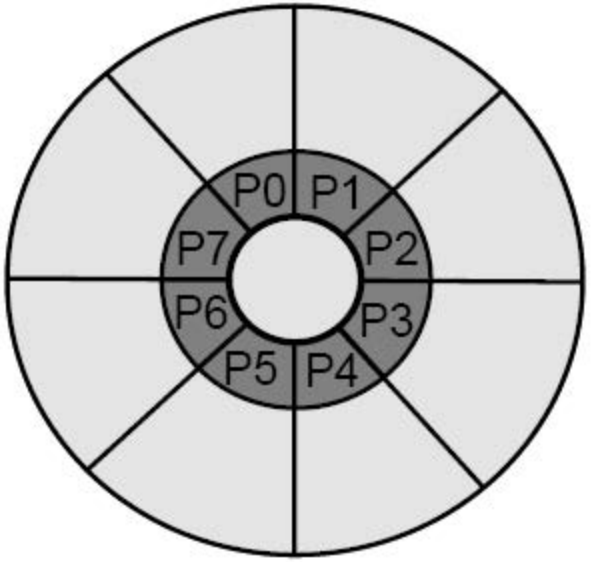
(d) Halo Constructed wit Non-uniform Size Banks



Nahalal

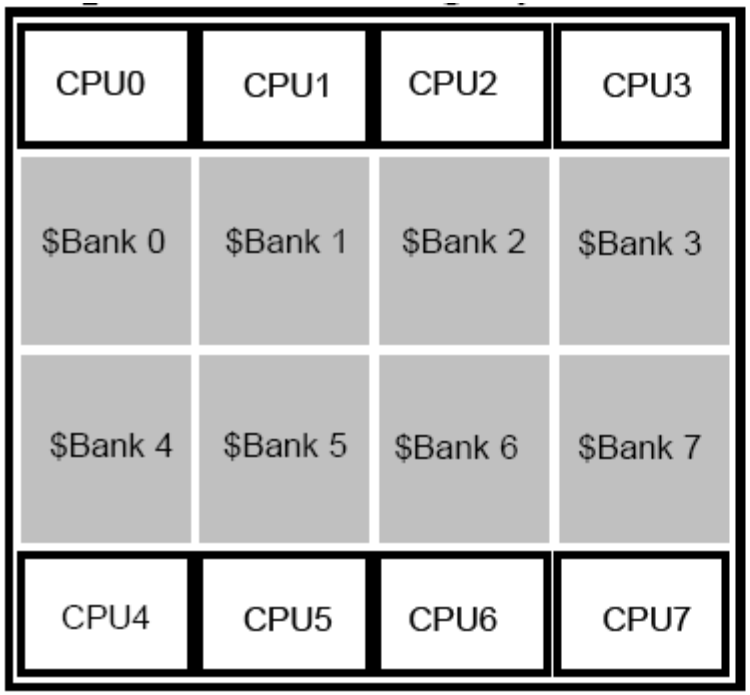
Guz et al., CAL'07

(a) Aerial view of Nahalal Village.

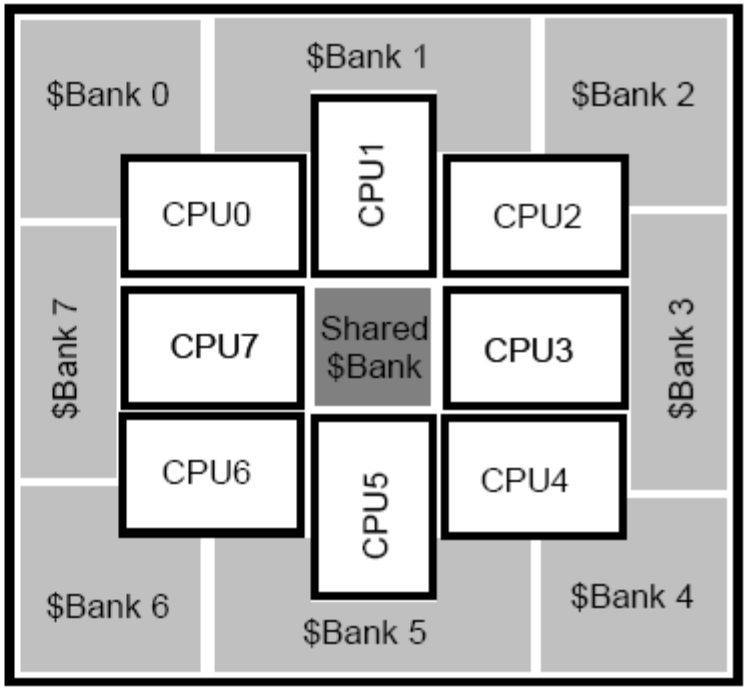


(b) CMP conceptual layout scheme.

Nahalal



(a) CIM layout.



(b) Nahalal layout.

- Block is initially placed in core's private bank and then swapped into the shared bank if frequently accessed by other cores
- Parallel search across all banks

Interconnection Networks

- Recall: fully connected network, arrays/rings, meshes/tori, trees, butterflies, hypercubes
- Consider a k-ary d-cube: a d-dimension array with k elements in each dimension, there are links between elements that differ in one dimension by 1 (mod k)
- Number of nodes $N = k^d$

(with no wraparound)

Number of switches	:	N
Switch degree	:	2d + 1
Number of links	:	Nd
Pins per node	:	2wd

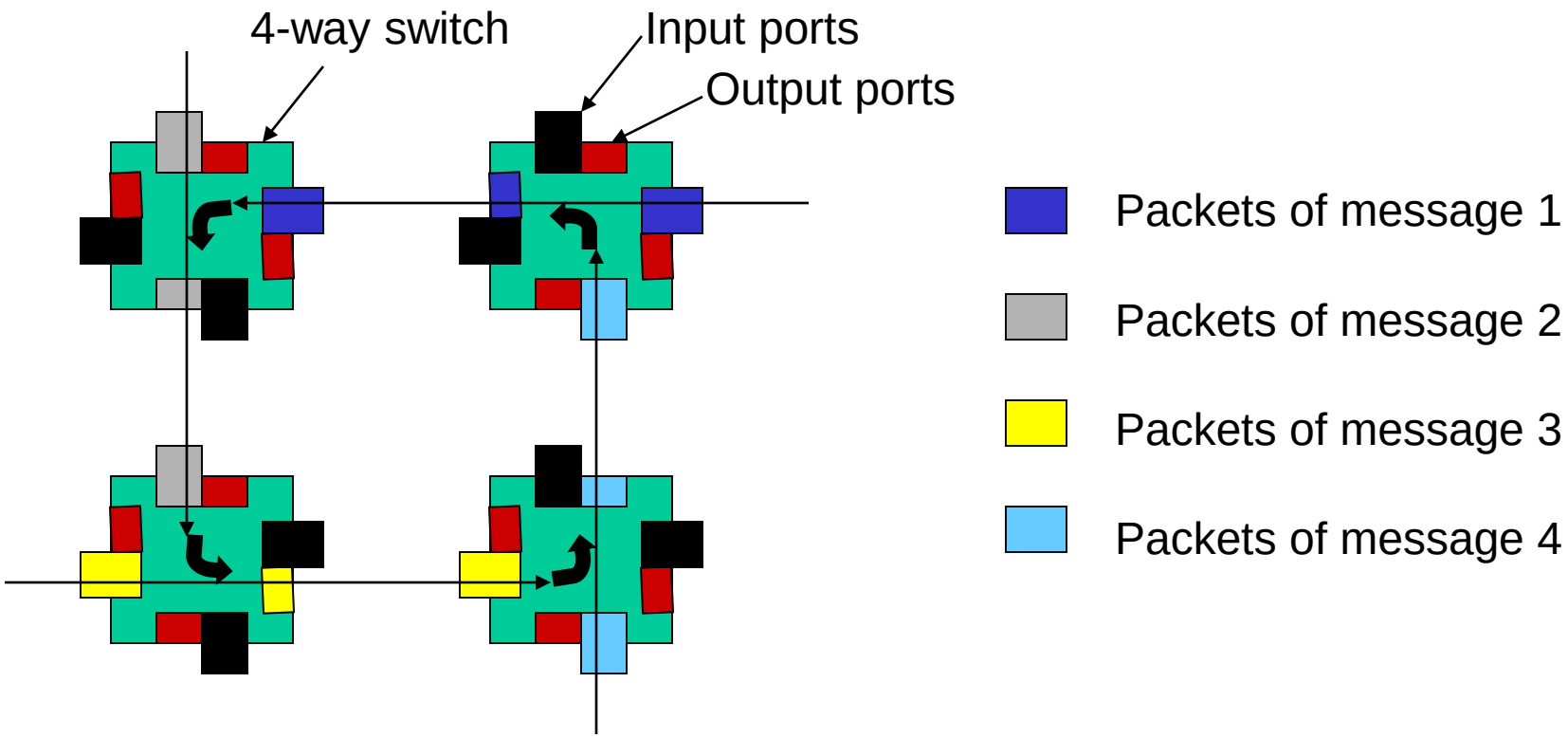
Avg. routing distance	:	d(k-1)/2
Diameter	:	d(k-1)
Bisection bandwidth	:	$2wk^{d-1}$
Switch complexity	:	$(2d + 1)^2$

Should we minimize or maximize dimension?

Routing

- Deterministic routing: given the source and destination, there exists a unique route
- Adaptive routing: a switch may alter the route in order to deal with unexpected events (faults, congestion) – more complexity in the router vs. potentially better performance
- Example of deterministic routing: dimension order routing: send packet along first dimension until destination co-ord (in that dimension) is reached, then next dimension, etc.

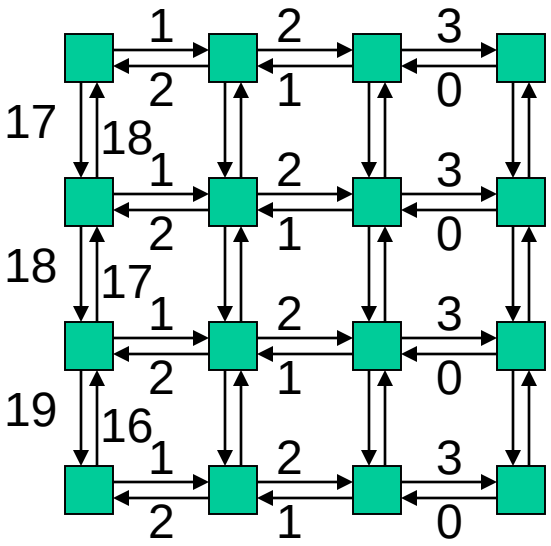
Deadlock Example



Each message is attempting to make a left turn – it must acquire an output port, while still holding on to a series of input and output ports

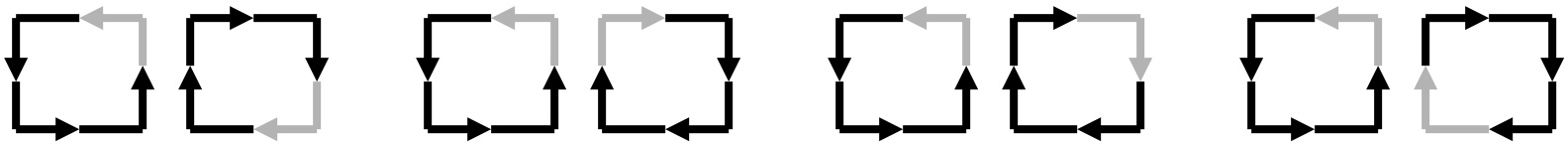
Deadlock-Free Proofs

- Number edges and show that all routes will traverse edges in increasing (or decreasing) order – therefore, it will be impossible to have cyclic dependencies
- Example: k-ary 2-d array with dimension routing: first route along x-dimension, then along y



Breaking Deadlock II

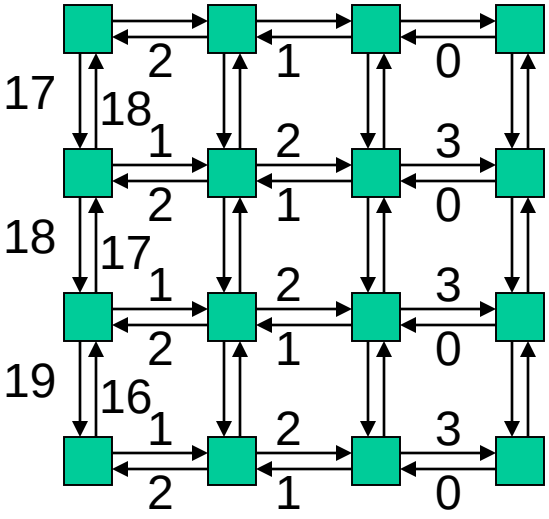
- Consider the eight possible turns in a 2-d array (note that turns lead to cycles)
- By preventing just two turns, cycles can be eliminated
- Dimension-order routing disallows four turns
- Helps avoid deadlock even in adaptive routing

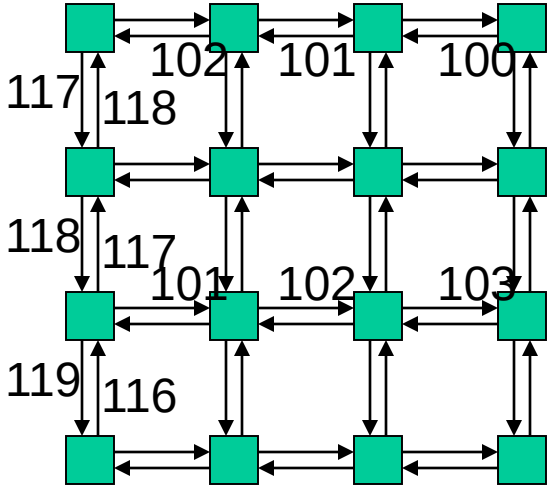


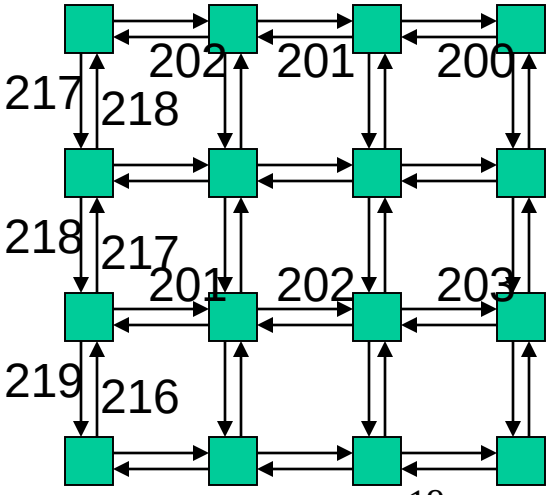
West-First | North-Last | Negative-First | Can allow deadlocks

Deadlock Avoidance with VCs

- VCs provide another way to number the links such that a route always uses ascending link numbers







- Alternatively, use West-first routing on the 1st plane and cross over to the 2nd plane in case you need to go West again (the 2nd plane uses North-last, for example)

Packets/Flits

- A message is broken into multiple packets (each packet has header information that allows the receiver to re-construct the original message)
- A packet may itself be broken into flits – flits do not contain additional headers
- Two packets can follow different paths to the destination
  Flits are always ordered and follow the same path
- Such an architecture allows the use of a large packet size (low header overhead) and yet allows fine-grained resource allocation on a per-flit basis

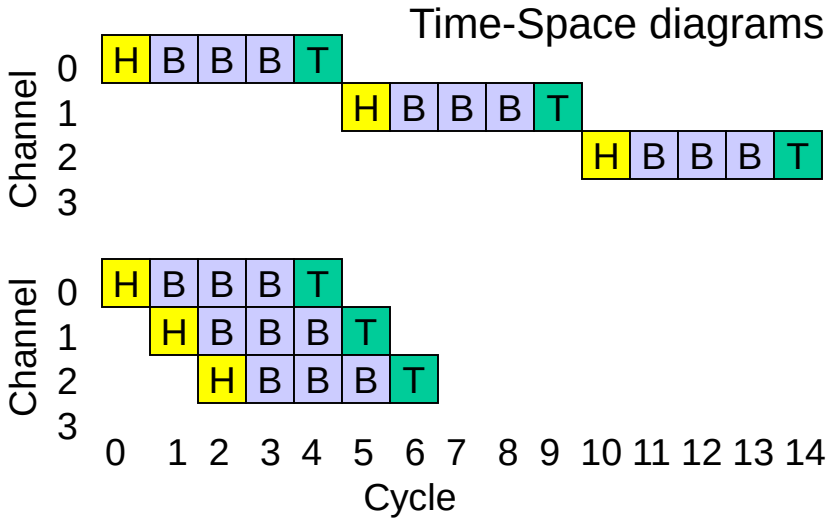
Flow Control

- The routing of a message requires allocation of various resources: the channel (or link), buffers, control state
- Bufferless: flits are dropped if there is contention for a link, NACKs are sent back, and the original sender has to re-transmit the packet
- Circuit switching: a request is first sent to reserve the channels, the request may be held at an intermediate router until the channel is available (hence, not truly bufferless), ACKs are sent back, and subsequent packets/flits are routed with little effort (good for bulk transfers)

Buffered Flow Control

- A buffer between two channels decouples the resource allocation for each channel – buffer storage is not as precious a resource as the channel (perhaps, not so true for on-chip networks)

- Packet-buffer flow control: channels and buffers are allocated per packet
  - Store-and-forward
  - Cut-through



Flit-Buffer Flow Control (Wormhole)

- Wormhole Flow Control: just like cut-through, but with buffers allocated per flit (not channel)

- A head flit must acquire three resources at the next switch before being forwarded:
  - channel control state (virtual channel, one per input port)
  - one flit buffer
  - one flit of channel bandwidth

  The other flits adopt the same virtual channel as the head and only compete for the buffer and physical channel

- Consumes much less buffer space than cut-through routing – does not improve channel utilization as another packet cannot cut in (only one VC per input port)

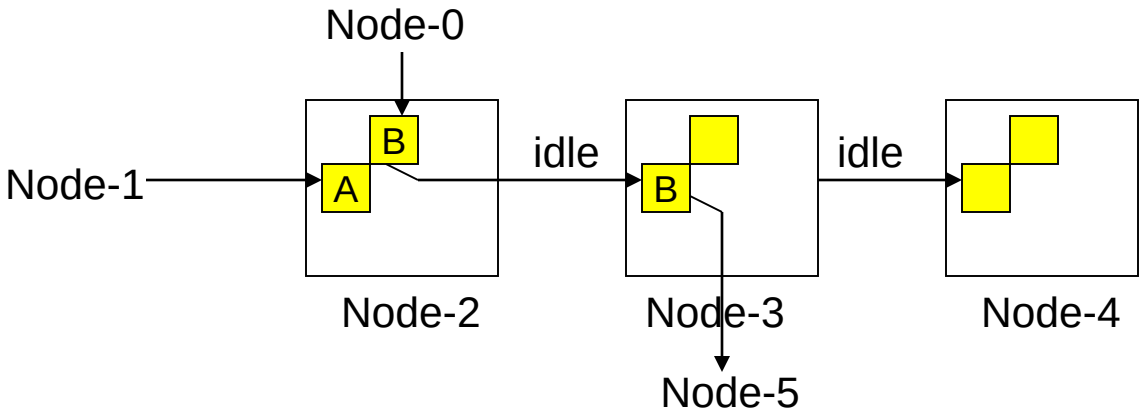
Virtual Channel Flow Control

- Each switch has multiple virtual channels per phys. channel
- Each virtual channel keeps track of the output channel assigned to the head, and pointers to buffered packets
- A head flit must allocate the same three resources in the next switch before being forwarded
- By having multiple virtual channels per physical channel, two different packets are allowed to utilize the channel and not waste the resource when one packet is idle

Example

- Wormhole:

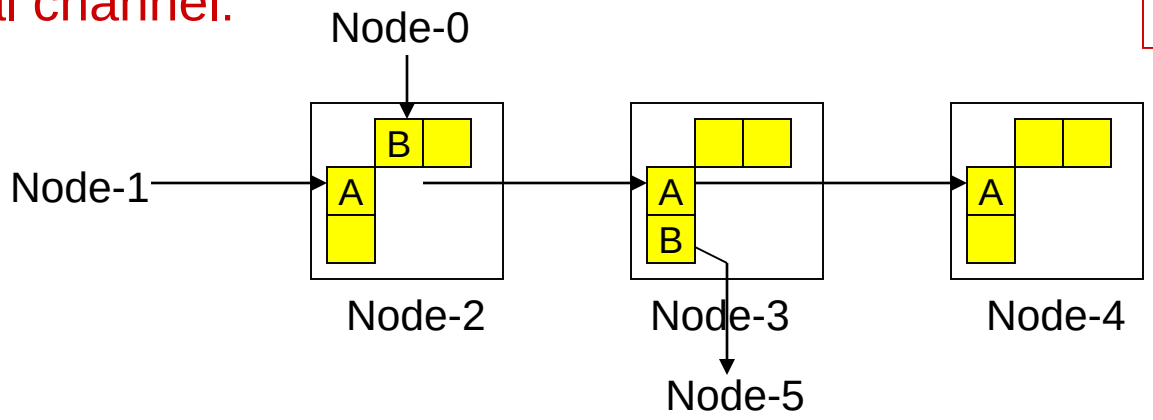
A is going from Node-1 to Node-4; B is going from Node-0 to Node-5



(blocked, no free VCs/buffers)

Traffic Analogy:
B is trying to make a left turn; A is trying to go straight; there is no left-only lane with wormhole, but there is one with VC

- Virtual channel:



(blocked, no free VCs/buffers)

Buffer Management

- Credit-based: keep track of the number of free buffers in the downstream node; the downstream node sends back signals to increment the count when a buffer is freed; need enough buffers to hide the round-trip latency

- On/Off: the upstream node sends back a signal when its buffers are close to being full – reduces upstream signaling and counters, but can waste buffer space

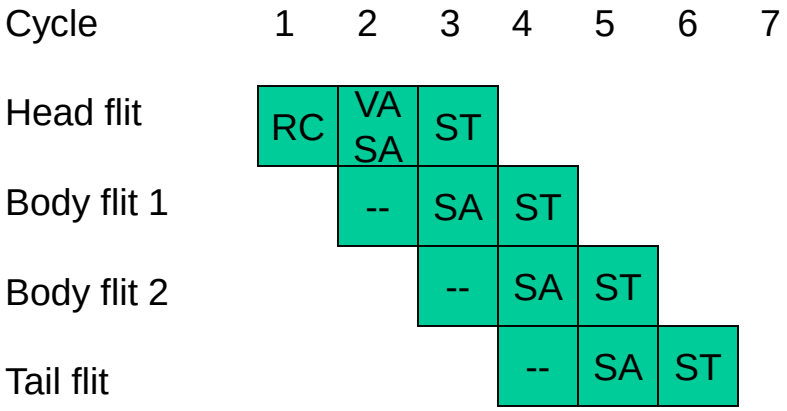
Router Pipeline

- Four typical stages:
  - RC routing computation: the head flit indicates the VC that it belongs to, the VC state is updated, the headers are examined and the next output channel is computed (note: this is done for all the head flits arriving on various input channels)
  - VA virtual-channel allocation: the head flits compete for the available virtual channels on their computed output channels
  - SA switch allocation: a flit competes for access to its output physical channel
  - ST switch traversal: the flit is transmitted on the output channel

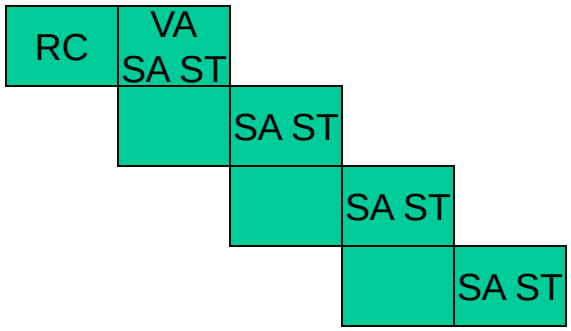
A head flit goes through all four stages, the other flits do nothing in the first two stages (this is an in-order pipeline and flits can not jump ahead), a tail flit also de-allocates the VC

Speculative Pipelines

- Perform VA and SA in parallel
- Note that SA only requires knowledge of the output physical channel, not the VC
- If VA fails, the successfully allocated channel goes un-utilized

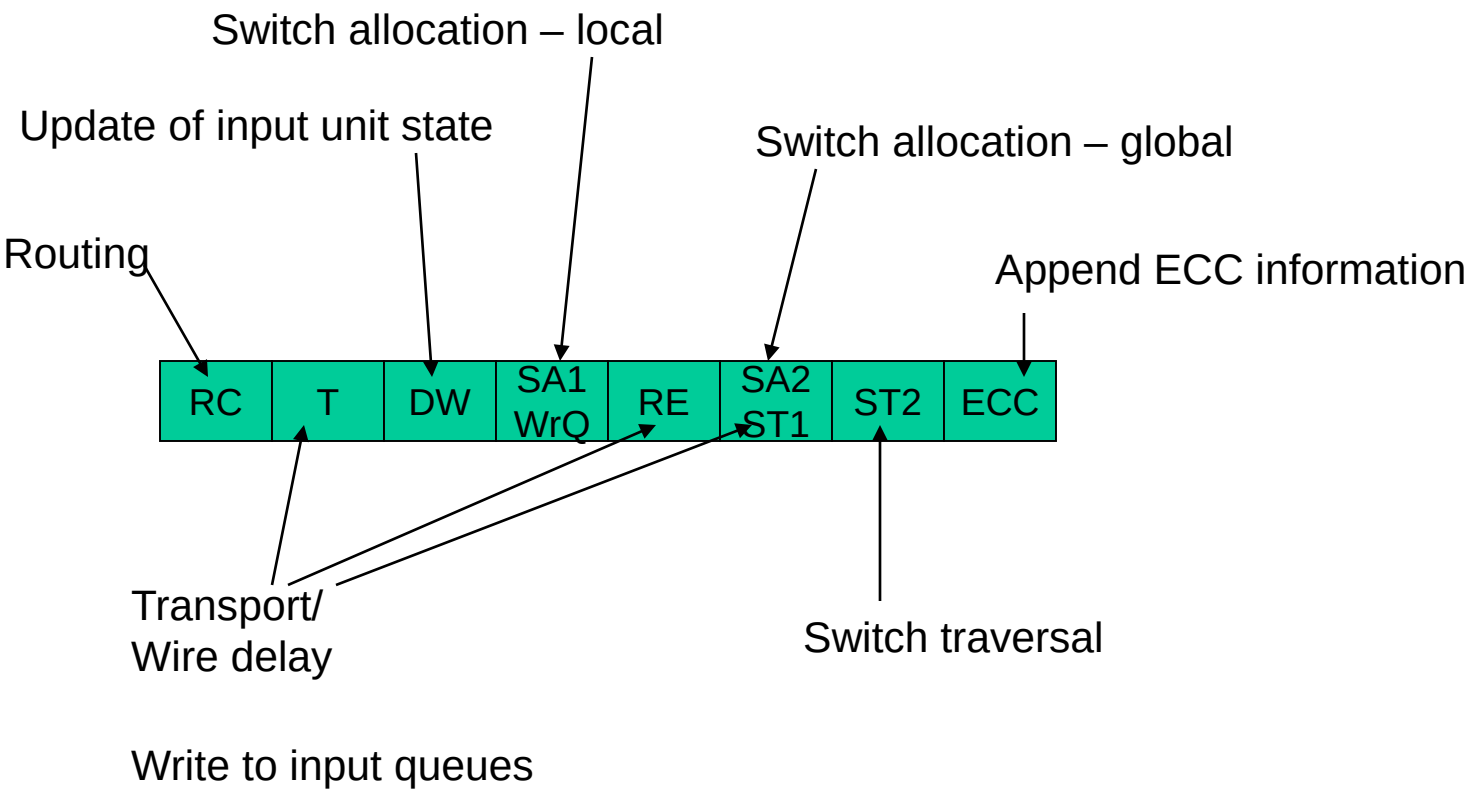


- Perform VA, SA, and ST in parallel (can cause collisions and re-tries)
- Typically, VA is the critical path – can possibly perform SA and ST sequentially

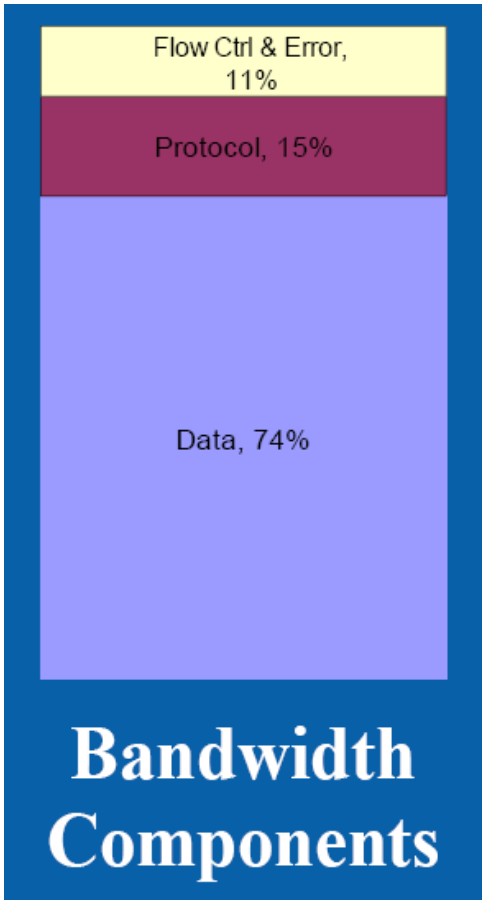


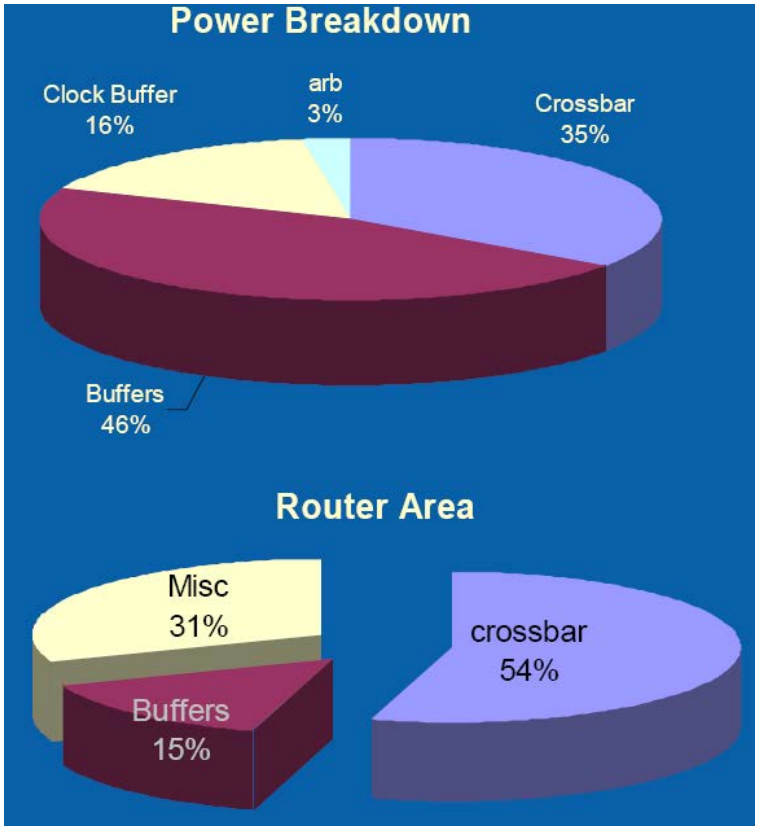
- Router pipeline latency is a greater bottleneck when there is little contention
- When there is little contention, speculation will likely work well!
- Single stage pipeline?

Alpha 21364 Pipeline



Recent Intel Router

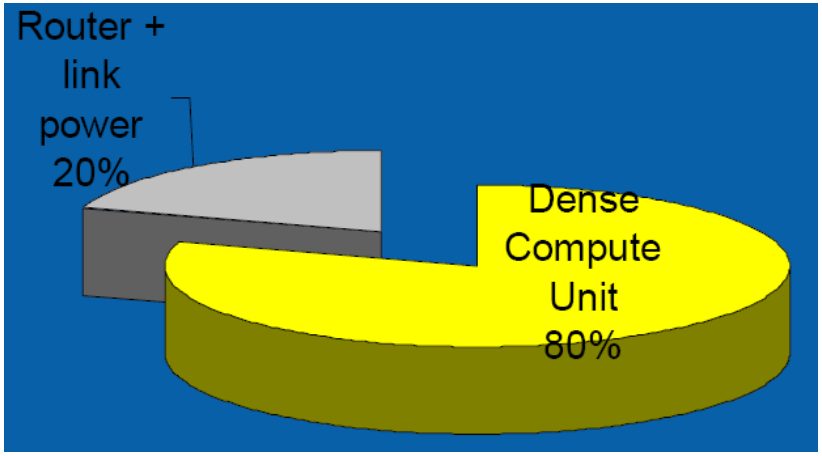


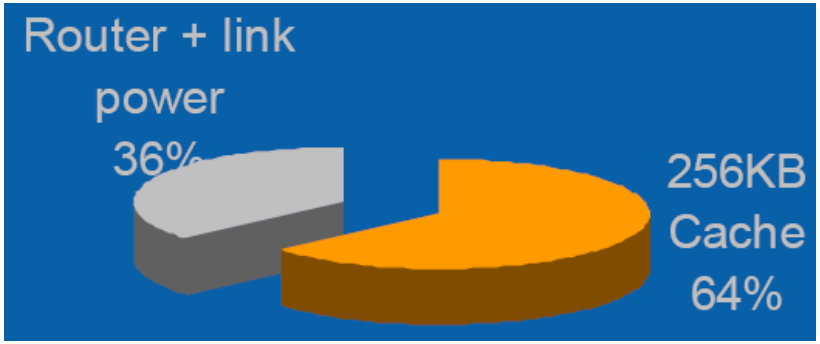


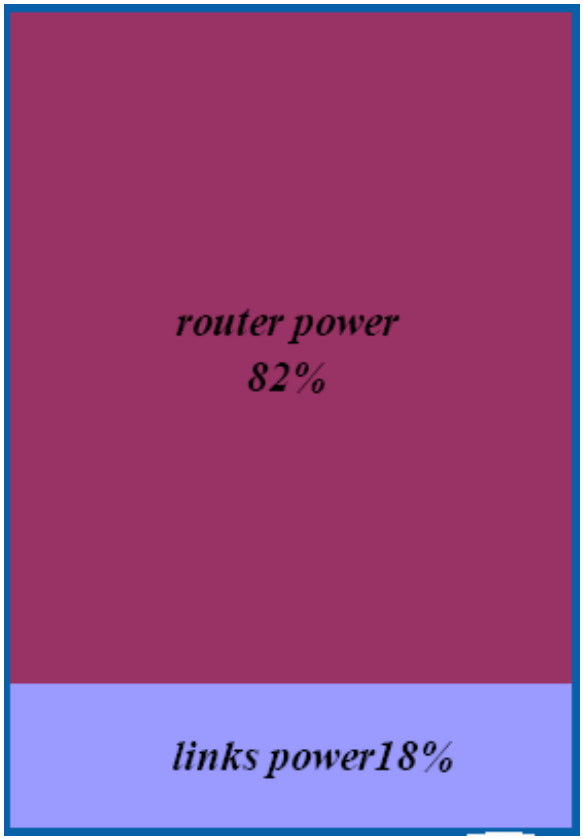
- Used for a 6x6 mesh
- 16 B, > 3 GHz
- Wormhole with VC flow control

Source: Partha Kundu, “On-Die Interconnects for Next-Generation CMPs”, talk at On-Chip Interconnection Networks Workshop, Dec 2006

Recent Intel Router

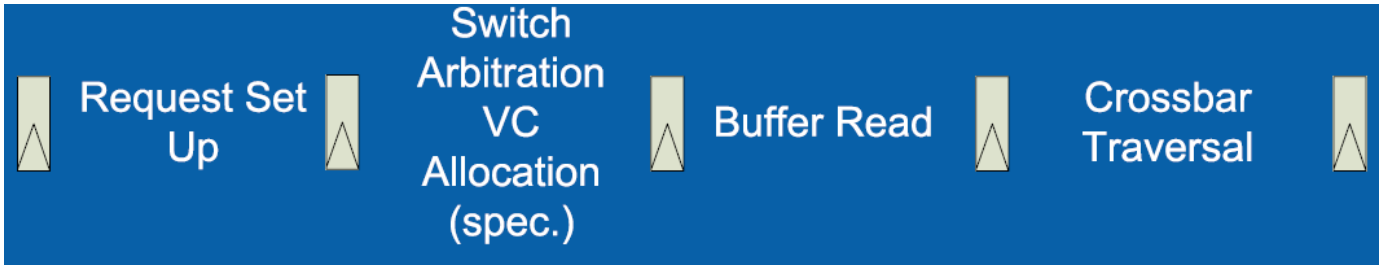


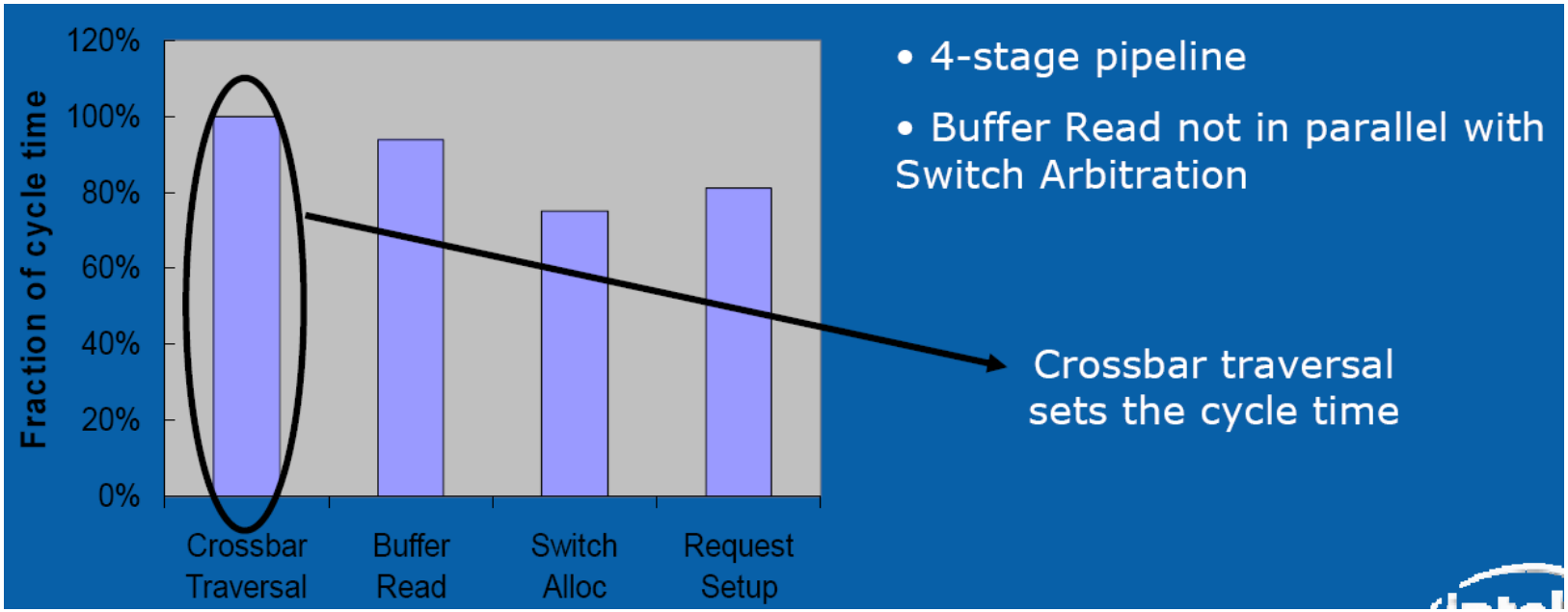




Source: Partha Kundu, “On-Die Interconnects for Next-Generation CMPs”, talk at On-Chip Interconnection Networks Workshop, Dec 2006

Recent Intel Router





Source: Partha Kundu, “On-Die Interconnects for Next-Generation CMPs”, talk at On-Chip Interconnection Networks Workshop, Dec 2006

Data Points

- On-chip network's power contribution
  - in RAW (tiled) processor: 36%
  - in network of compute-bound elements (Intel): 20%
  - in network of storage elements (Intel): 36%
  - bus-based coherence (Kumar et al. '05): ~12%
  - Polaris (Intel) network: 28%
  - SCC (Intel) network: 10%

- Power contributors:
  - RAW: links 39%; buffers 31%; crossbar 30%
  - TRIPS: links 31%; buffers 35%; crossbar 33%
  - Intel: links 18%; buffers 38%; crossbar 29%; clock 13%

Title

- Bullet