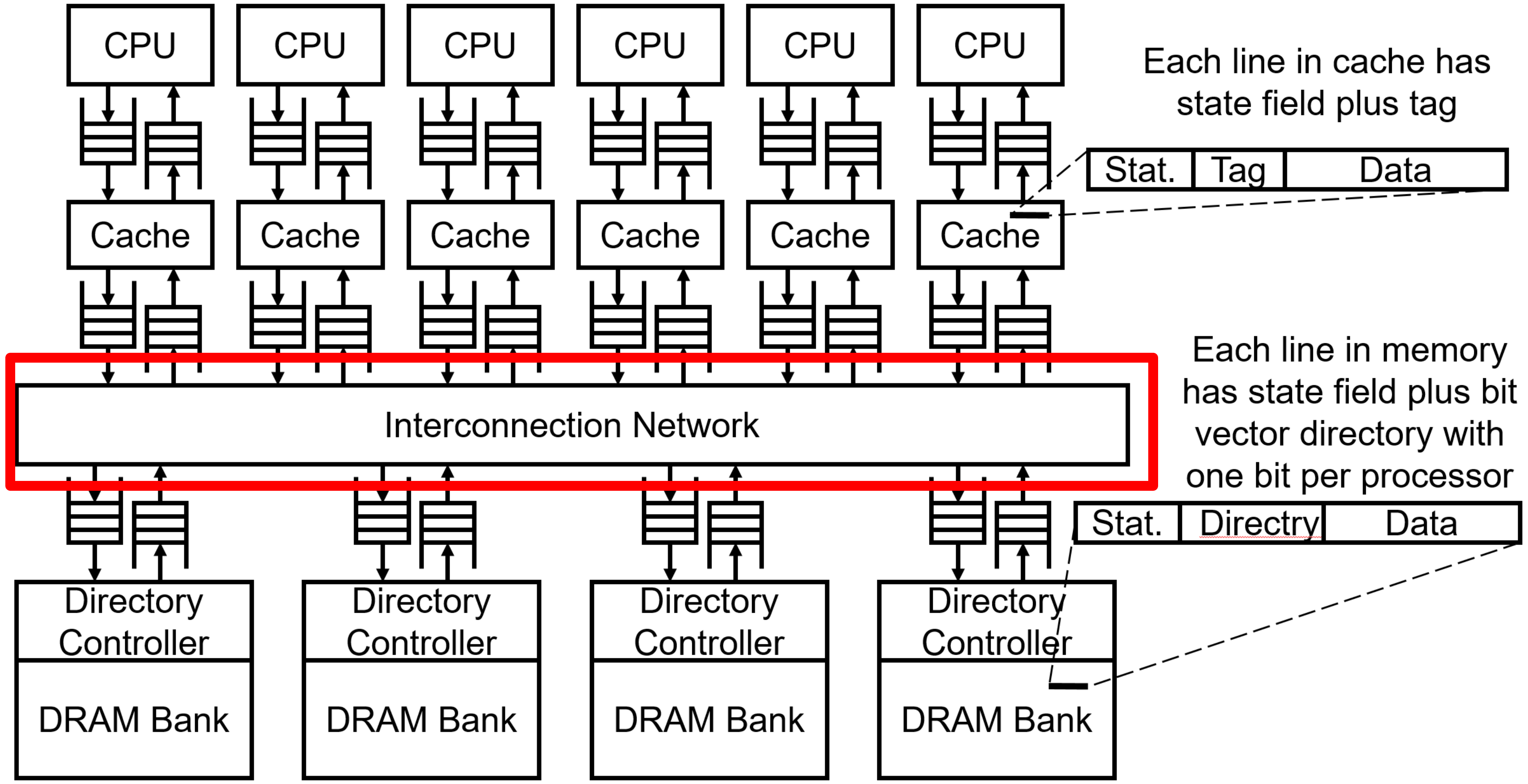
CS 152 Computer Architecture and Engineering
CS252 Graduate Computer Architecture

Lecture 25 - Interconnects

Jerry Zhao
Electrical Engineering and Computer Sciences
University of California at Berkeley

jzh@berkeley.edu

Directory Cache Protocol



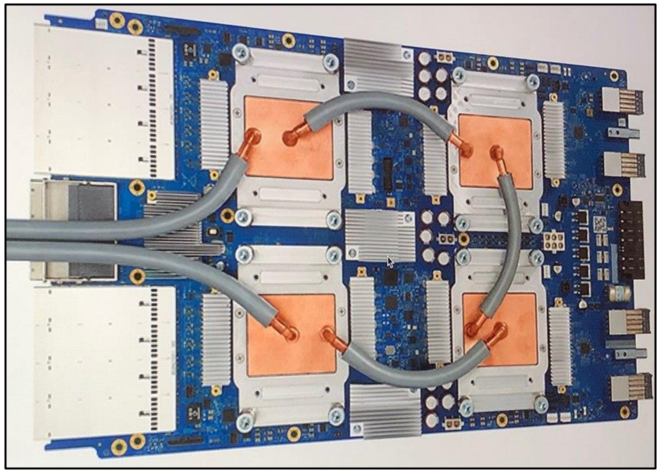
CPU
Cache
Interconnection Network
Directory Controller
DRAM Bank
Each line in cache has state field plus tag
Stat.
Tag
Data
Each line in memory has state field plus bit vector directory with one bit per processor
Stat.
Directry
Data

This Lecture

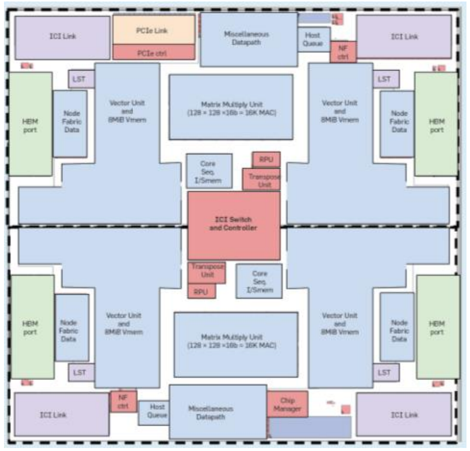
- Interconnects are not traditionally covered in 152
  - More in-depth in EECS251B
- This Lecture:
  - Topology
  - Flow-control
  - Routing
  - Micro-architecture
  - Notable interconnect implementations

Interconnect Scale

Rack-to-rack (datacenter)



Chip-to-chip



On-chip

- This lecture will focus on **on-chip interconnects for shared-memory systems**
- Same principles apply when scaling out

Interconnect Agents

Agents on a **shared-memory interconnect** can be:

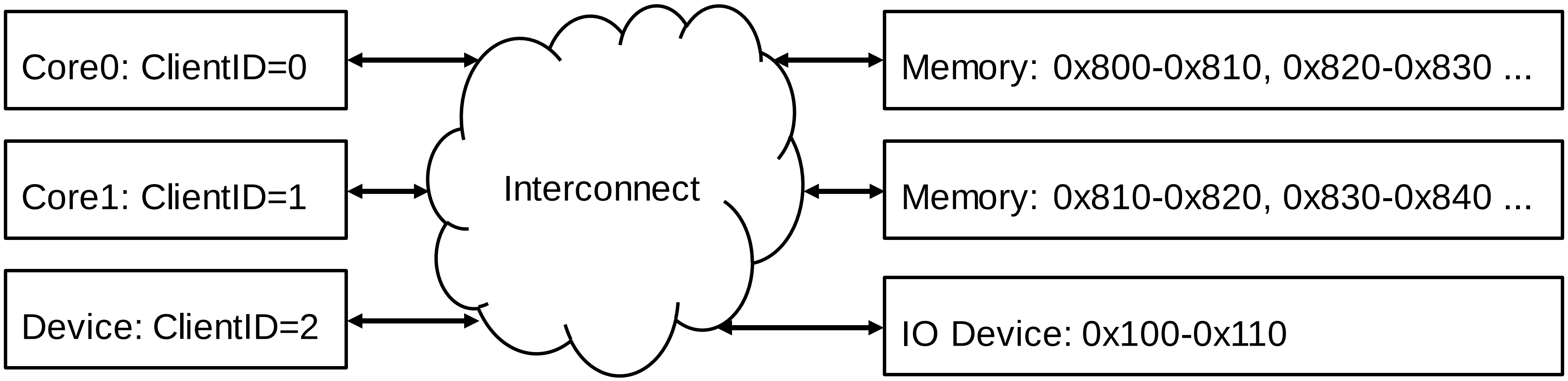
- Cores/private-caches
- Shared-caches/memory banks
- Accelerators
- I/O devices (ethernet, PCI-E, HDMI)
- Off-chip memory channels (DDR)

Alternative interpretations:

- FPGA routing between CLBs
- Datapath between PEs in an accelerator

Architectural View

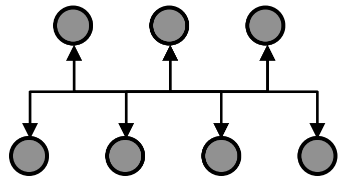
- Interconnect agents speak a common memory protocol
- AMBA/AXI/ACE/TileLink/Wishbone
- Interconnect implementation routes traffic to correct destination
  - Client-to-manager traffic: route by address
  - Manager-to-client traffic: route by client identifier (ID)
- Across homogeneous channels/banks, address ranges can be striped or hashed for load-balancing



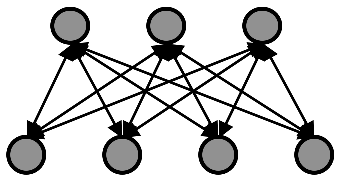
Interconnect Components

- **Topology** – physical layout of routers (nodes) and channels (links)
- **Flow control** – mechanics of how messages travel from source-to-destination
- **Routing** – which paths messages take through the network
- **Micro-architecture** – implementation of routers and channels

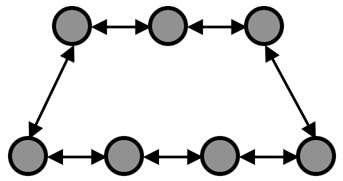
Topologies



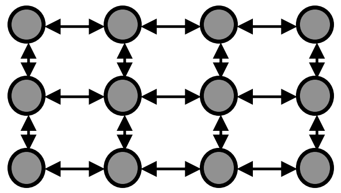
- Shared-bus
  - Advantages: cheap & easy to implement, broadcast, serialized messages
  - Disadvantages: low bandwidth, tri-state logic



- Crossbar – all-to-all connection
  - Advantages: high bandwidth due to all-to-all routing, no contention, serialized messages, predictable latency
  - Disadvantages: O(n^2) scaling, scales poorly past 4x4 networks



- **1D torus/ring (unidirectional/bidirectional)**
  - Advantages: simple to implement, well-behaved
  - Disadvantages: low bisection bandwidth, high-hop-count



- **2D mesh**
  - Advantages: scalable with good bandwidth/low-latency
  - Disadvantages: complex routing for deadlock-freedom

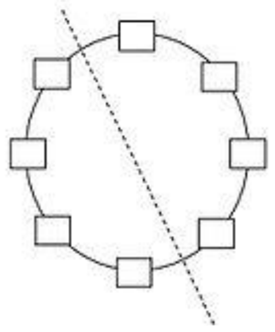
These are **network-on-chips (NoCs),**
with proper routers and channels

Topology Properties

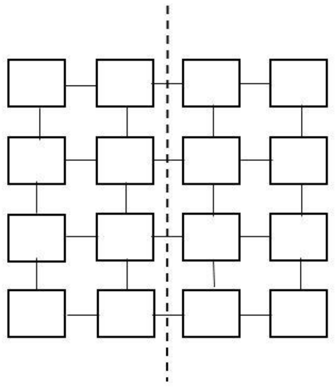
- Diameter – longest shortest path between any two nodes
- Bisection bandwidth – minimum bandwidth between any two partitions

Assuming single-direction channel bandwidth is **b:**

- 1D unidirectional torus with **n** nodes
  - Diameter = n - 1
  - Bisection bandwidth = 2b

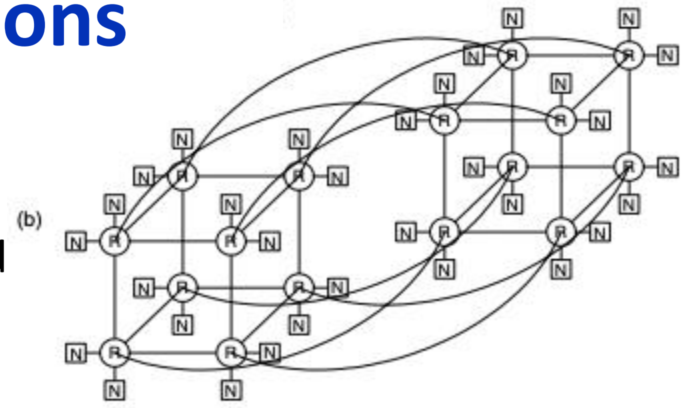


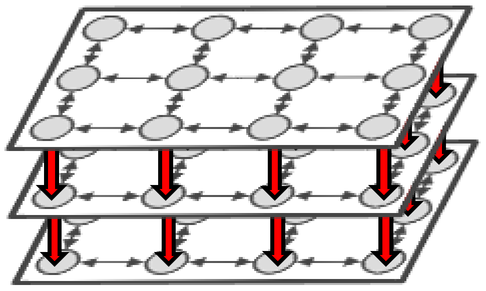
- 1D bidirectional torus with **n** nodes:
  - Diameter = n / 2
  - Bisection bandwidth = 4b
- **n x n** 2D mesh
  - Diameter = 2n
  - Bisection bandwidth = 2nb

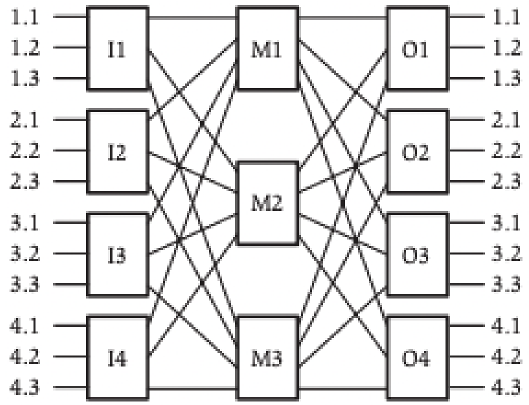


Topology Variations

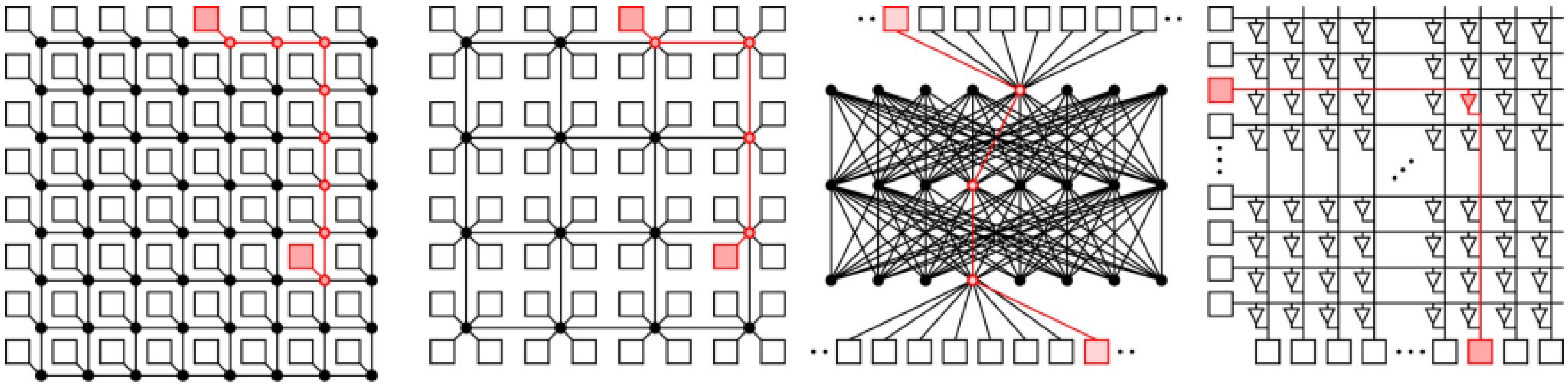
- Hypercube
  - Popular in the 70s/80s, connect multi-board machines
  - Low diameter, high bandwidth
  - Simpler 1D/2D topologies won for single-chip multi-processors
- 3D mesh
  - Extension of 2D mesh
  - 3D integration with TSVs (through-silicon-vias)
- Butterfly/Clos networks
  - Multi-stage crossbar
  - Used in big network switches

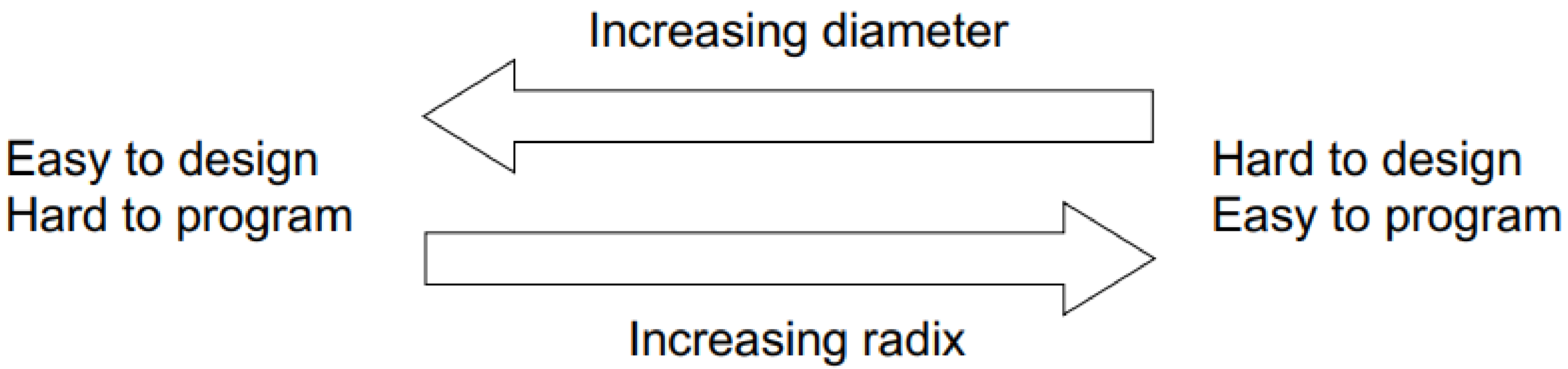






Topology Spectrum



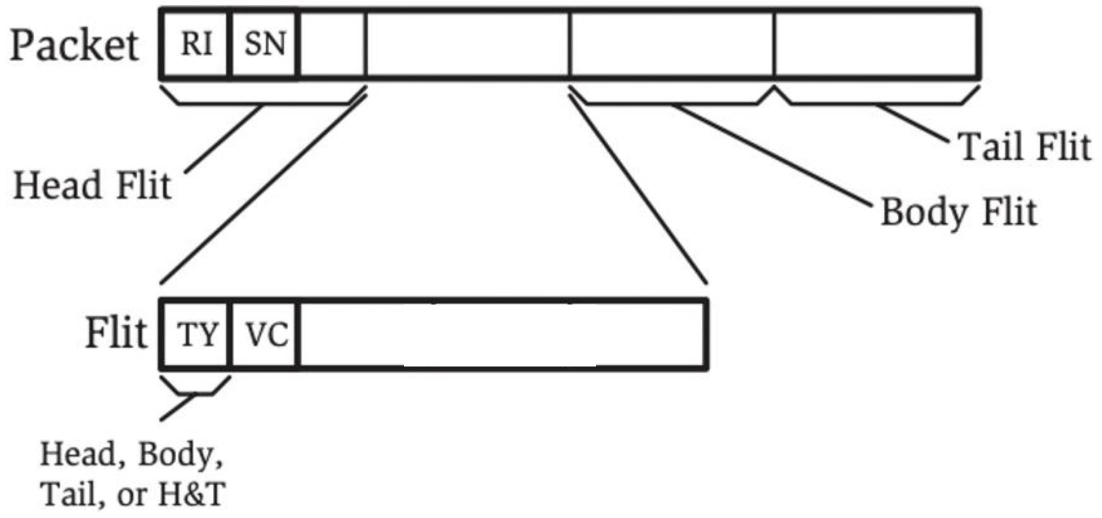


Radix – Number of inputs and outputs of each switching node
Diameter – largest minimal hop count over all node pairs

Flow Control – Packets/Flits

Packet: unit of routing/flow-control, a single message in the network

- Size can be from ~40b (just address) to ~600b (addr + cache line) to 1KB (streaming data)



Packets are subdivided into equal-length **flits,**

- **Head/tail flits** mark the boundaries of a packet
- All flits follow the same path in order
- Variable packet size => variable flits/packet

Flow Control – Packet/Flit Tradeoff

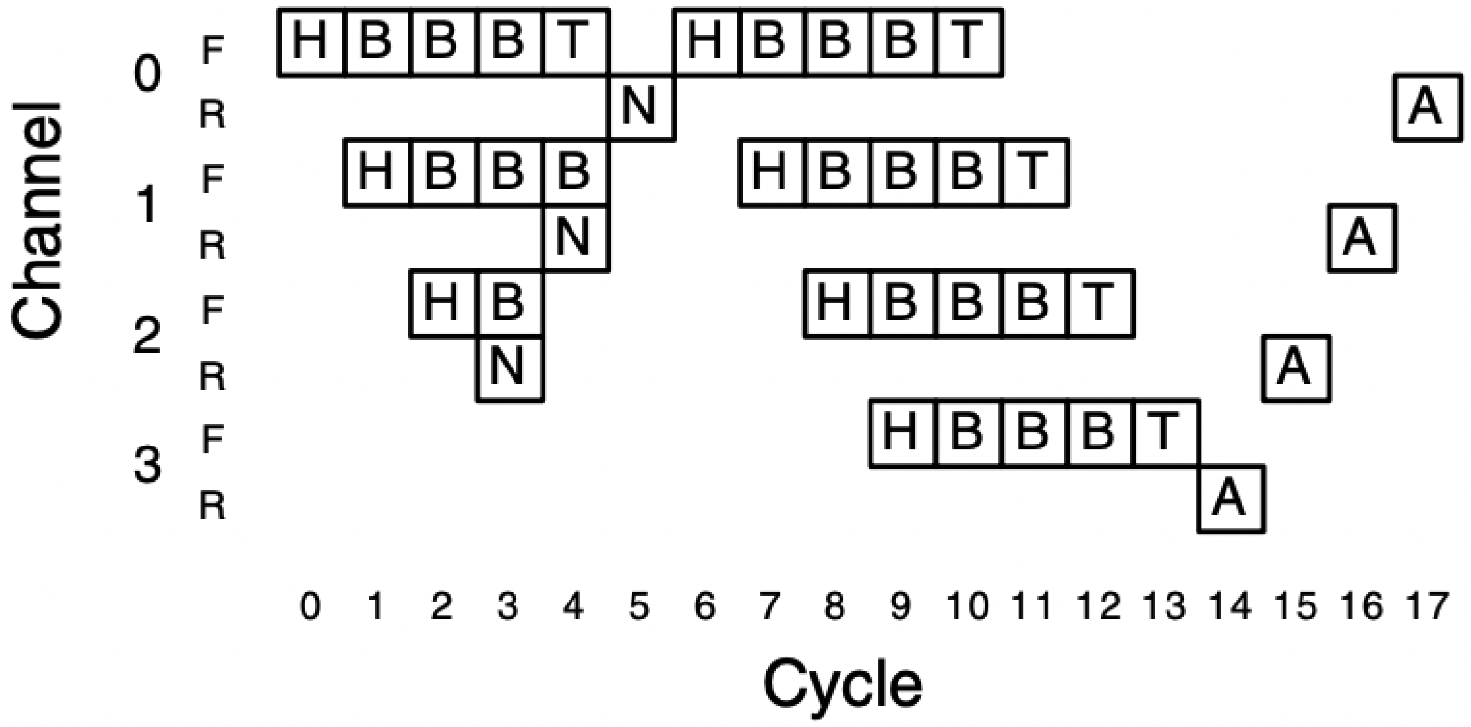
- Small packets/flits => better resource utilization
- Large packets/flits => reduce overheads

Flow Control – Two Types

- **Bufferless** : no buffers between ingress and egress, packets move straight from channel-to-channel
  - Like a train system/railroad
- **Buffered:** packets can stall in a buffer in the network while they wait for a channel
  - Like a road system with intersections
  - Lanes are buffers

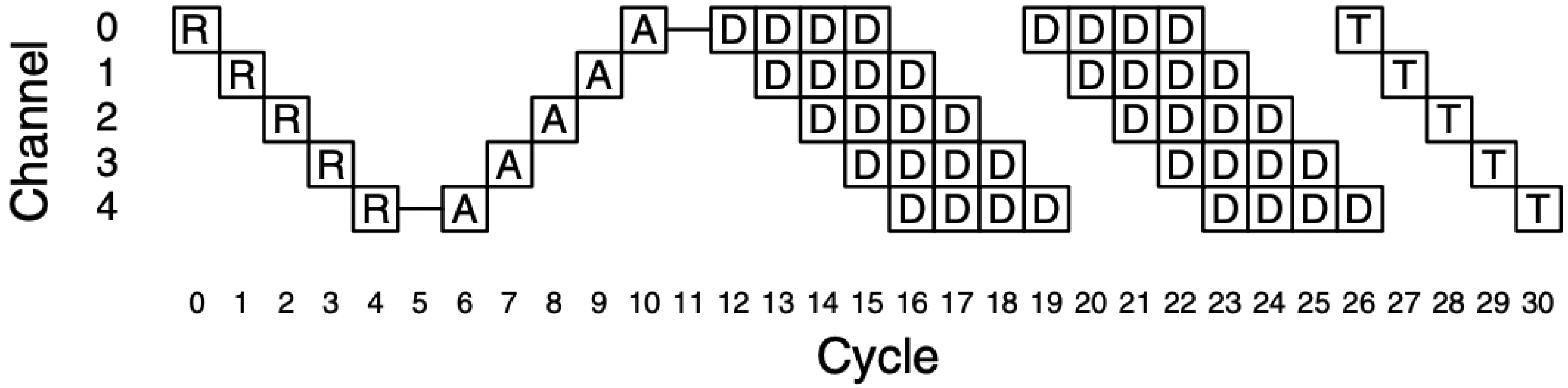
Flow Control - Dropping

- **Dropping flow control:** nack packets if they fail to allocate a channel, sender should retry
  - Sender must buffer all inflight requests
  - Low throughput under load, requires buffers



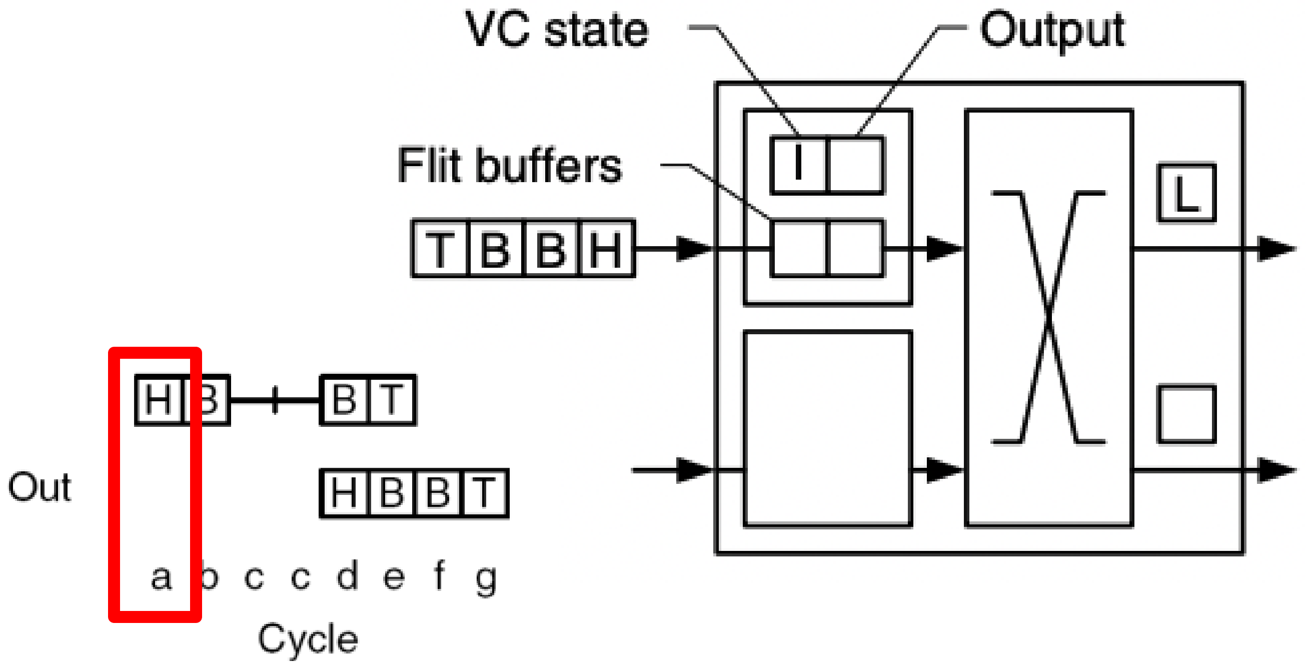
Flow Control – Circuit-switched

- **Circuit-switched flow control:** try to acquire the entire path from ingress to egress, only send packet if acquired
  - Low throughput/high latency
  - Telephone network works like this



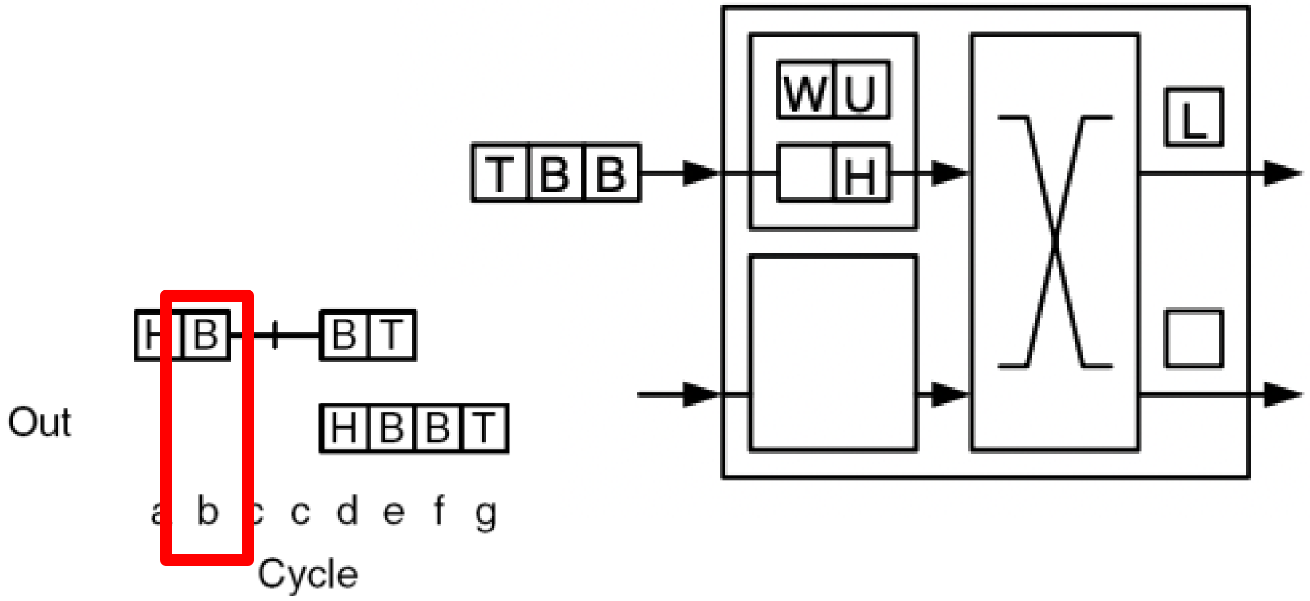
Flow Control - Buffered

- **Packet-buffered:** Buffer the entire packet, then send over the link
  - High buffer capacity requirements
- **Flit-buffered "wormhole" flow-control:** Buffer flits, not packets
  - Flits travel as a "worm" from head to tail



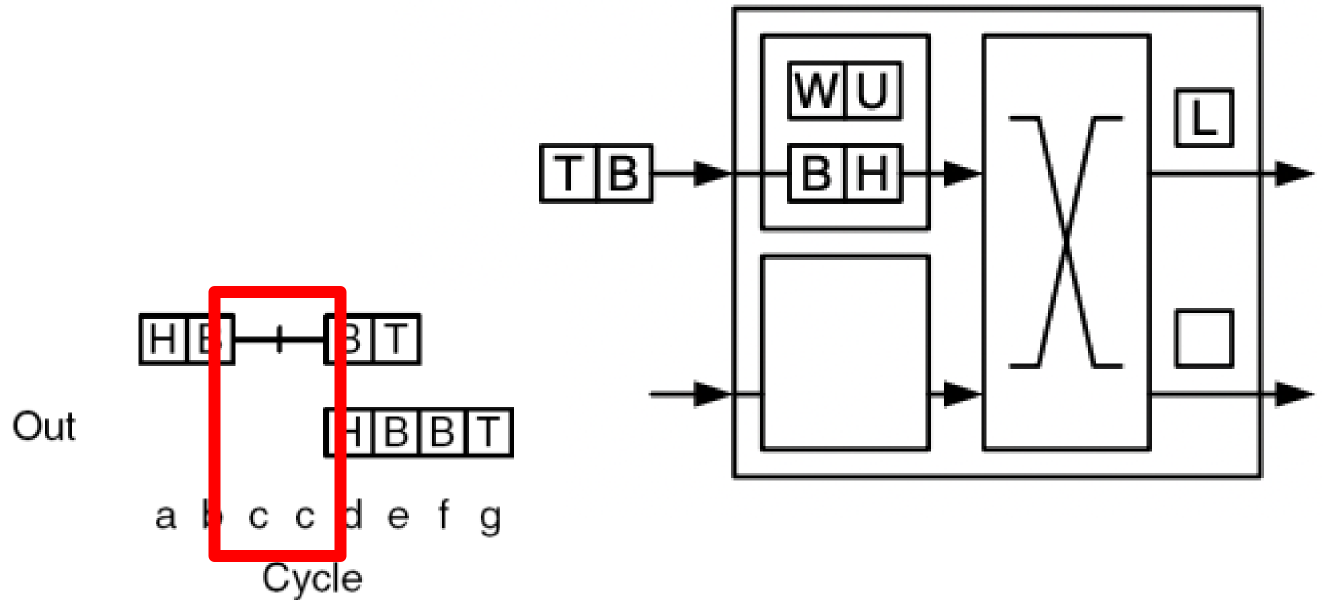
Flow Control - Buffered

- **Packet-buffered:** Buffer the entire packet, then send over the link
  - High buffer capacity requirements
- **Flit-buffered "wormhole" flow-control:** Buffer flits, not packets
  - Flits travel as a "worm" from head to tail



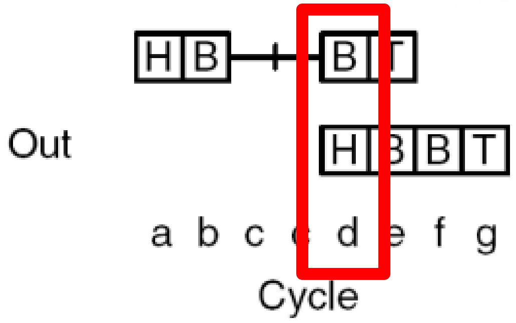
Flow Control - Buffered

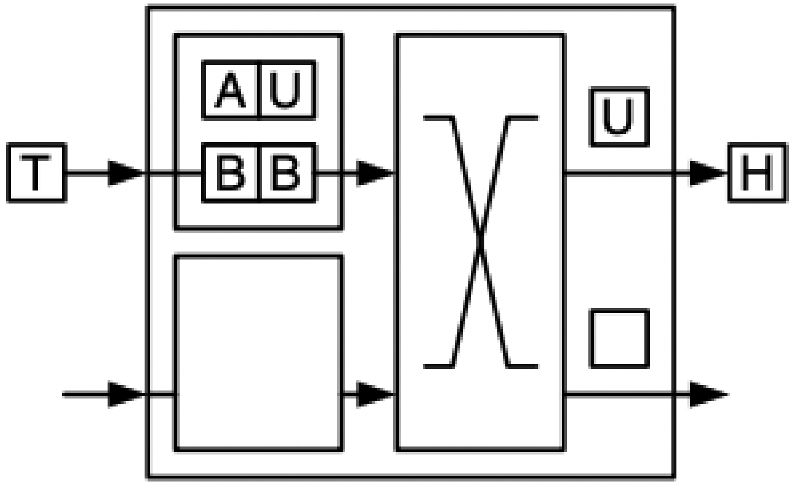
- **Packet-buffered:** Buffer the entire packet, then send over the link
  - High buffer capacity requirements
- **Flit-buffered "wormhole" flow-control:** Buffer flits, not packets
  - Flits travel as a "worm" from head to tail



Flow Control - Buffered

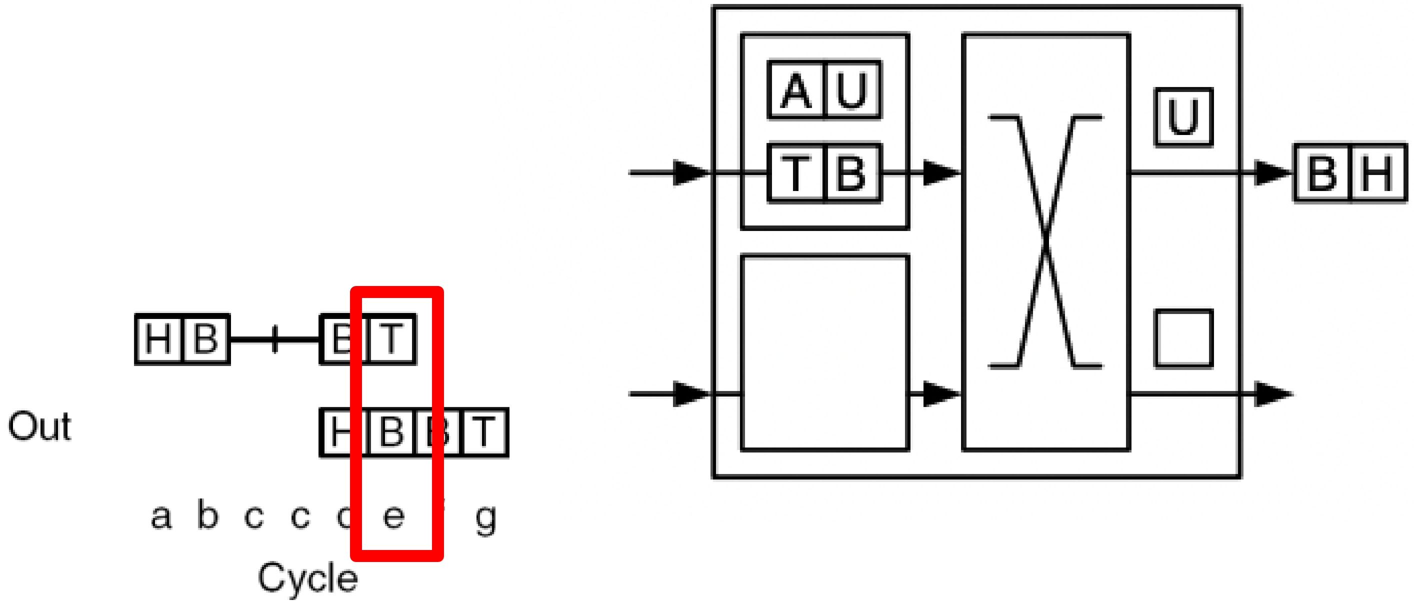
- **Packet-buffered:** Buffer the entire packet, then send over the link
  - High buffer capacity requirements
- **Flit-buffered "wormhole" flow-control:** Buffer flits, not packets
  - Flits travel as a "worm" from head to tail





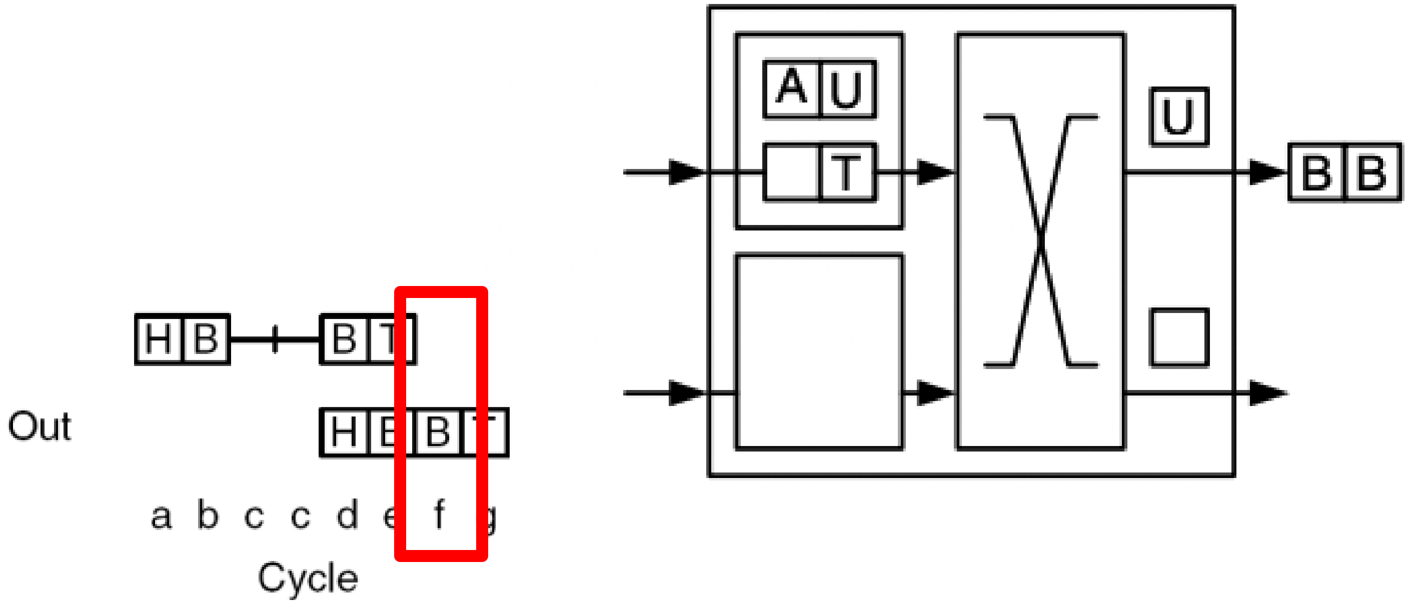
Flow Control - Buffered

- **Packet-buffered:** Buffer the entire packet, then send over the link
  - High buffer capacity requirements
- **Flit-buffered "wormhole" flow-control:** Buffer flits, not packets
  - Flits travel as a "worm" from head to tail



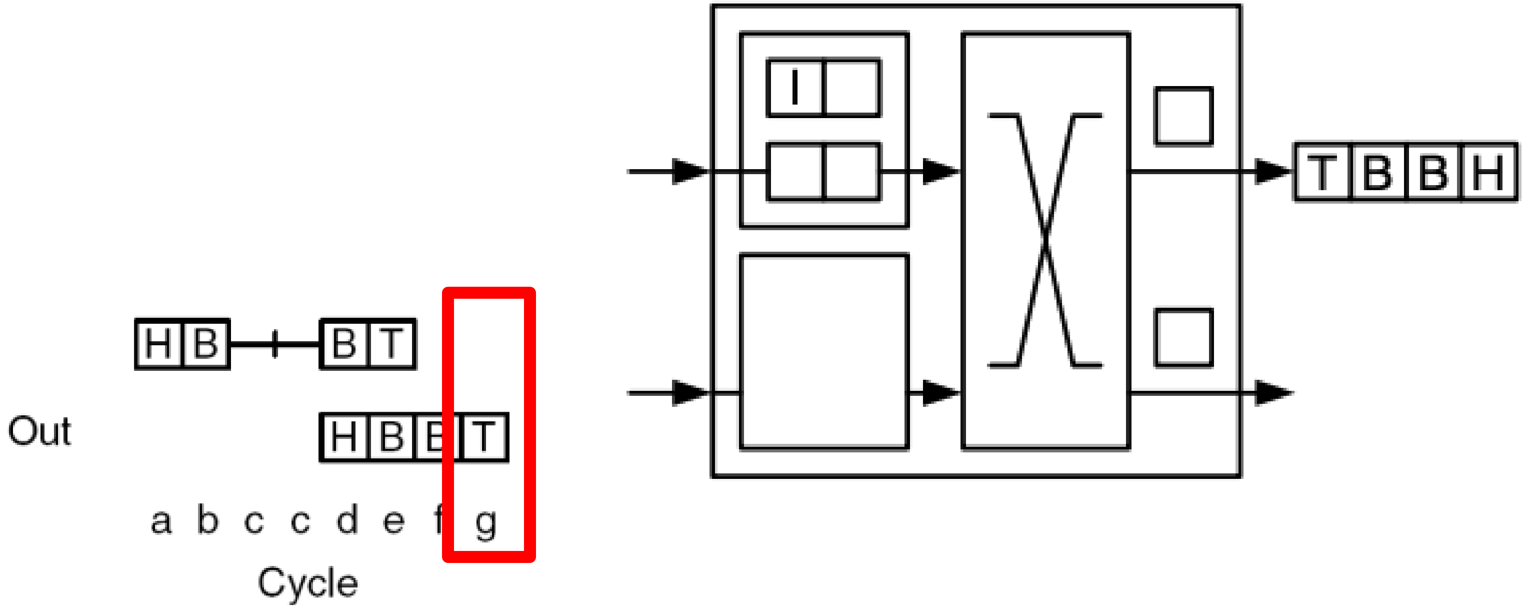
Flow Control - Buffered

- **Packet-buffered:** Buffer the entire packet, then send over the link
  - High buffer capacity requirements
- **Flit-buffered "wormhole" flow-control:** Buffer flits, not packets
  - Flits travel as a "worm" from head to tail



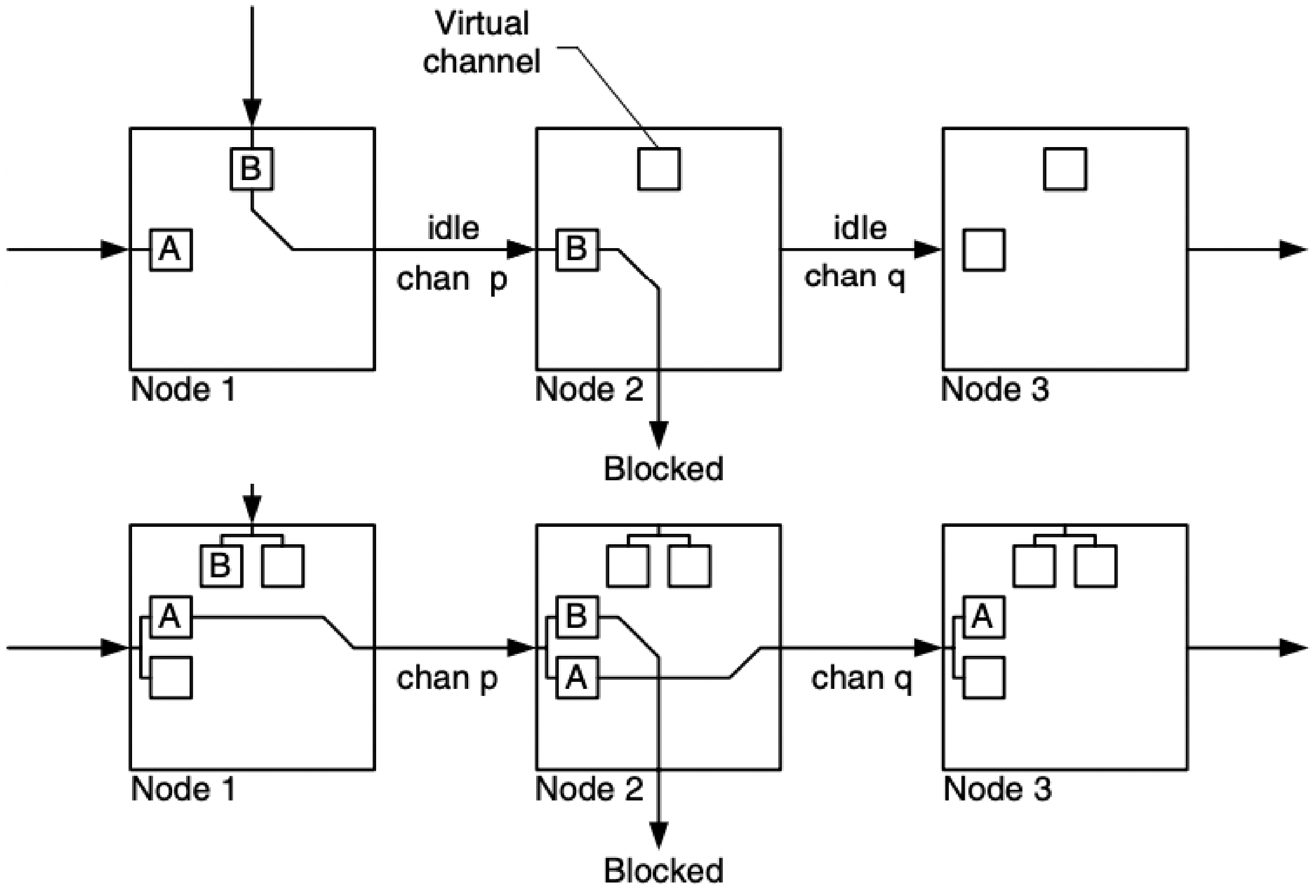
Flow Control - Buffered

- **Packet-buffered:** Buffer the entire packet, then send over the link
  - High buffer capacity requirements
- **Flit-buffered "wormhole" flow-control:** Buffer flits, not packets
  - Flits travel as a "worm" from head to tail



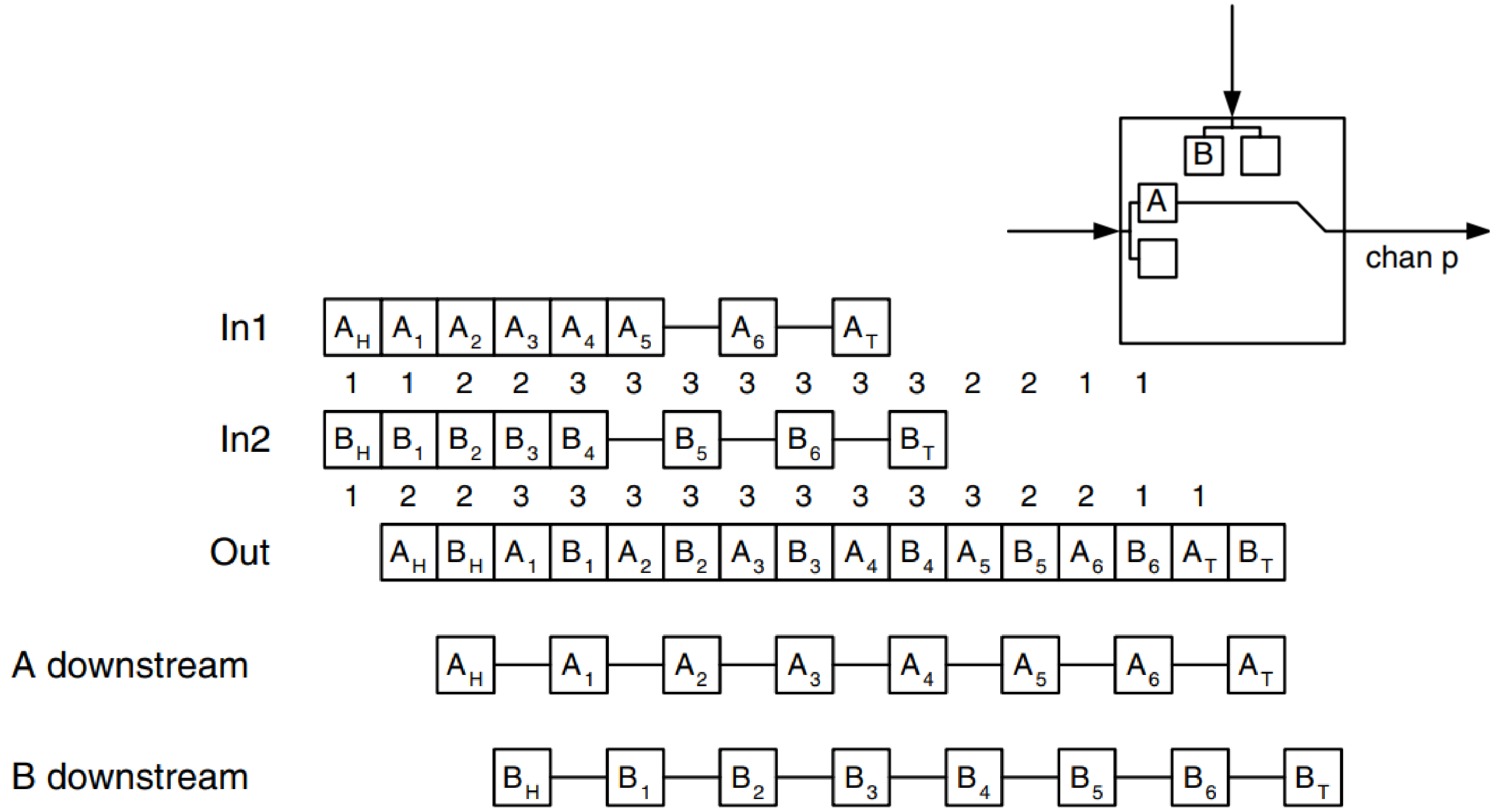
Flow Control – Virtual Channels

- Allow flits from different packets to **time-multiplex** access to a physical channel
- **Flit buffers are "virtual channels" (VCs)**
- Standard approach for NoC architectures



Flow Control – Virtual Channels

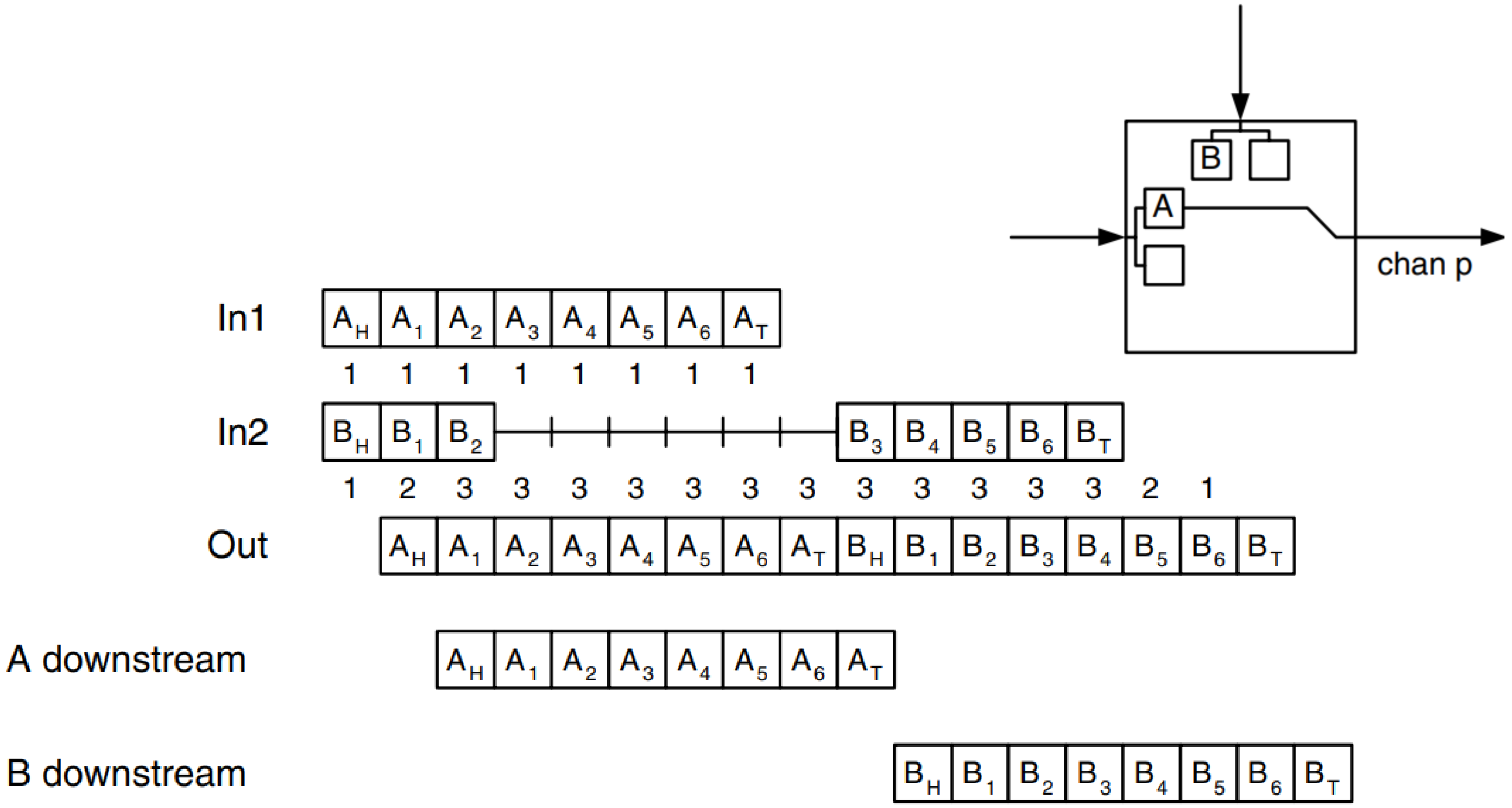
- Suppose A/B are contending for a single physical channel
- What happens if fair arbitration is used between **A flits** and **B flits**?



B
A
chan p
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
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

Flow Control – Virtual Channels

- Suppose A/B are contending for a single physical channel
- Winner-take-all arbitration improves latency

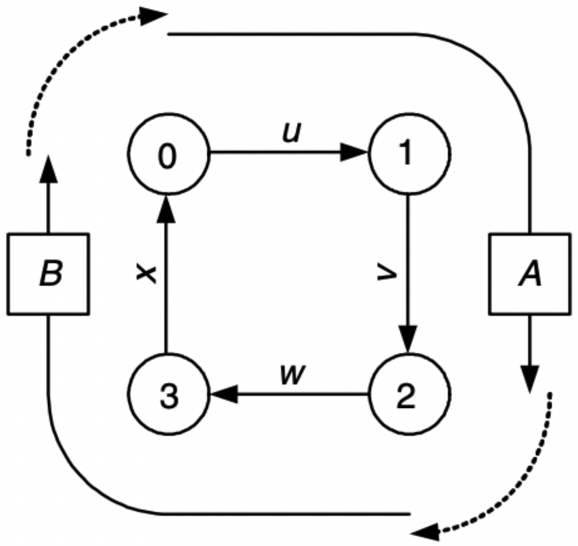


B
A
chan p
In1
A_H A_1 A_2 A_3 A_4 A_5 A_6 A_T
1 1 1 1 1 1 1 1
In2
B_H B_1 B_2 B_3 B_4 B_5 B_6 B_T
1 2 3 3 3 3 3 3 3 3 3 3 3 3 2 1
Out
A_H A_1 A_2 A_3 A_4 A_5 A_6 A_T B_H B_1 B_2 B_3 B_4 B_5 B_6 B_T
A downstream
A_H A_1 A_2 A_3 A_4 A_5 A_6 A_T
B downstream
B_H B_1 B_2 B_3 B_4 B_5 B_6 B_T

Routing

- **Deterministic** – no consideration for other traffic
  - Easy to implement, can fully calculate route at the source
  - Poor load-balancing
- **Adaptive** – consider existing traffic in the network
  - Consider network's current state
  - Can't query the entire network before routing, need to compute the route on-the-fly
- **Minimal** – route along minimal paths only

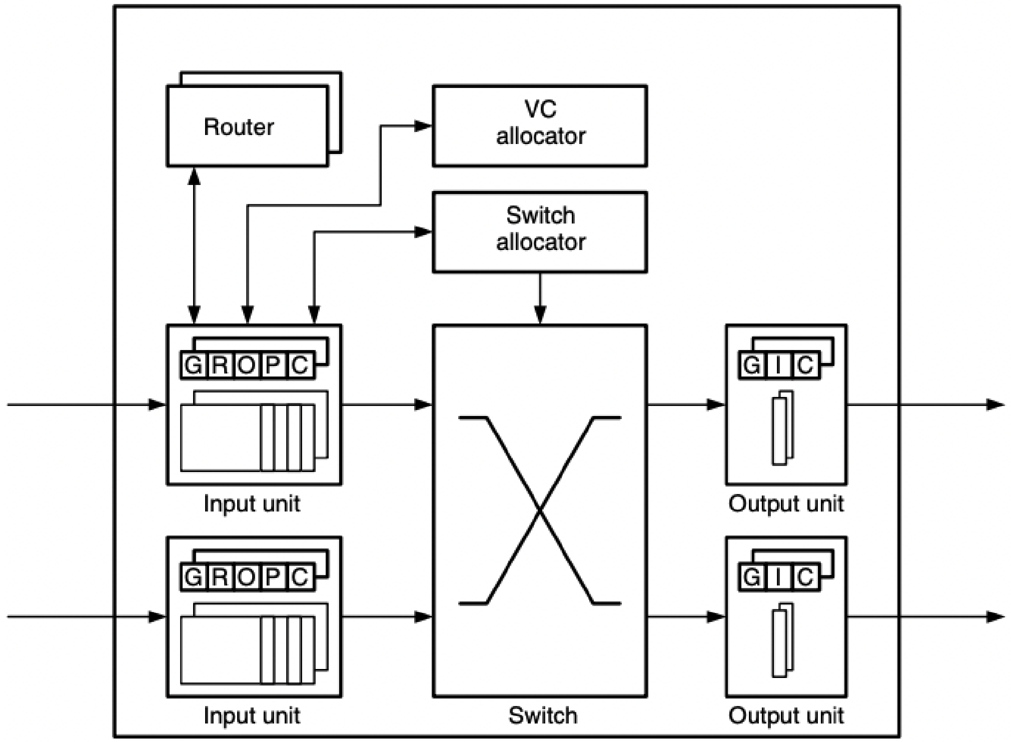
Deadlock-free routing



- Generalizes to more complex topologies – 2D mesh
- **Solution A**: break circular dependencies in channel dependency graph
  - Disadvantage: Requires lots of VCs (buffers)
- **Solution B:** disallow certain physical routes
  - Dimension ordered routing: route first in X, then Y
  - Disadvantage: Creates network hotspots
- Both A and B are used in complex routing algorithms

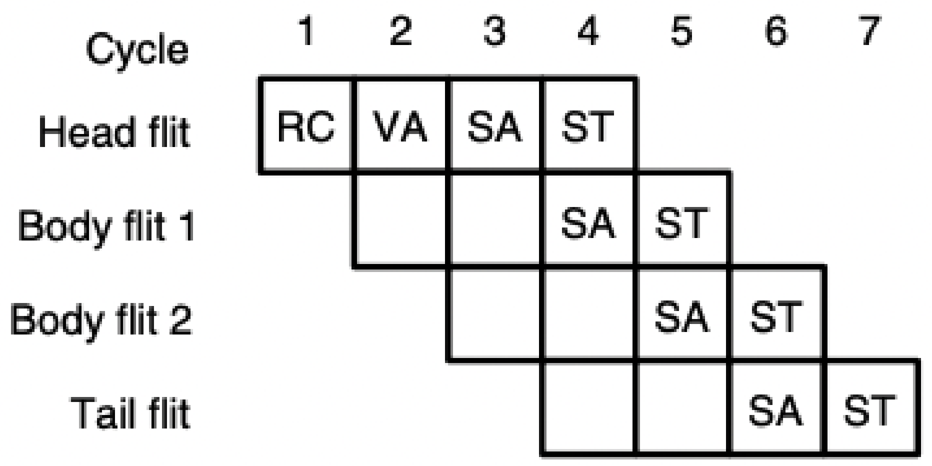
Router Microarchitecture

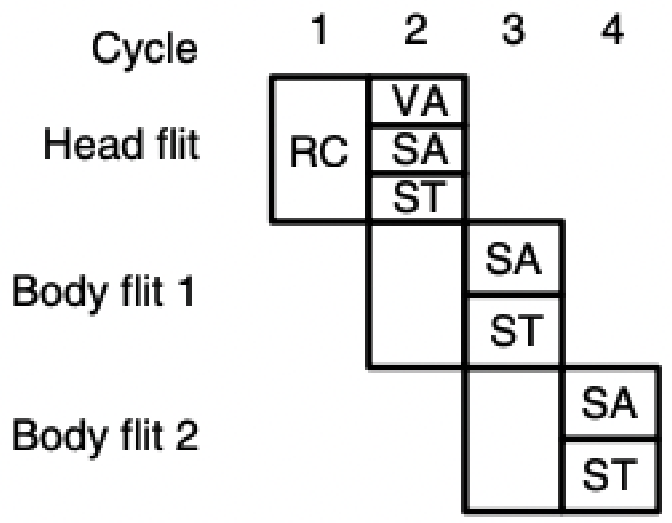
- Virtual channels (buffers)
- Allocators/arbiters schedule access to channels, buffers, switch
- Small crossbar-switch to connect inputs with outputs



Router Pipelining

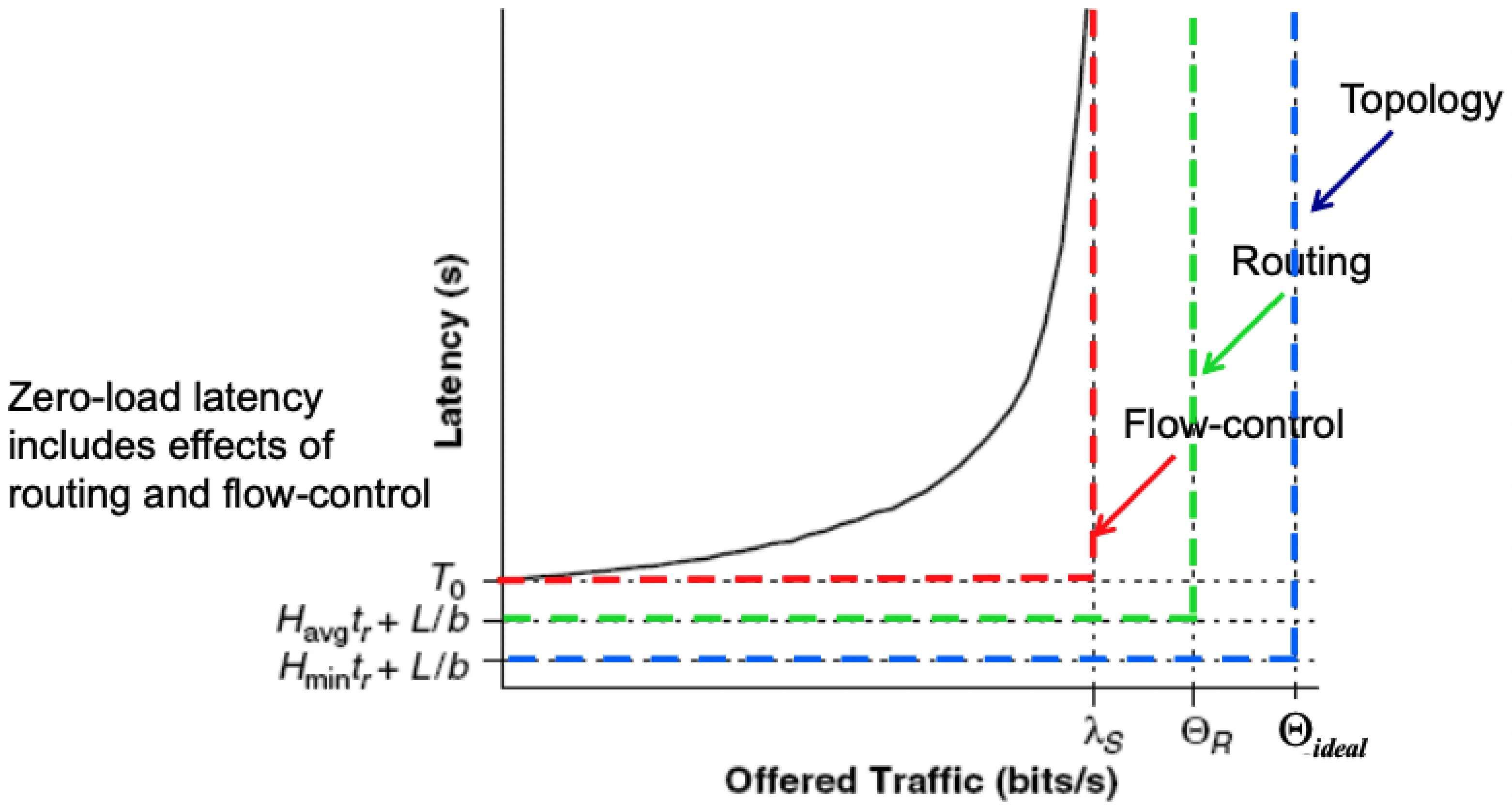
- Hop latency is critical, interconnect latency might be on the critical path for a L1/L2 miss
    - 1 cycle on a hop, is 2 cycles round trip
- Modern NoCs are 1-2 cycles per hop
    - Speculation allows merging of pipeline stages





Evaluating Interconnects

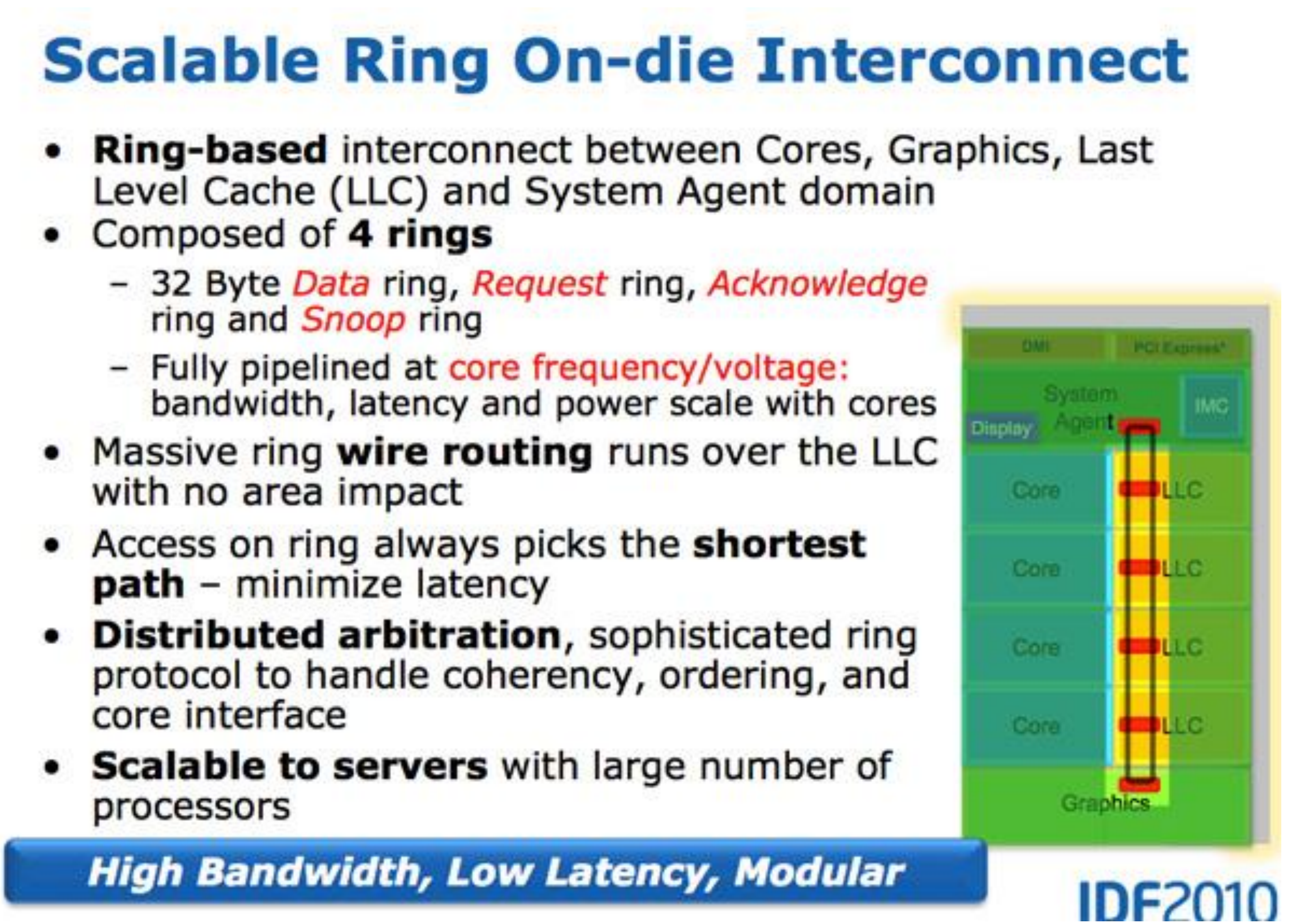
- Latency vs Offered Traffic
  - Measure average latency as across offered traffic
  - Traffic limit is the network capacity
  - Latency limit is zero-load latency



QoS – Quality of Service

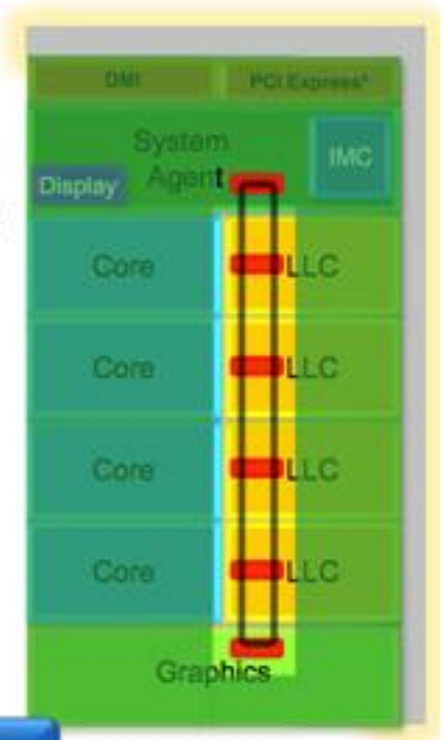
- **Service classes** – categorization of traffic by properties
  - **Real time** – camera input
  - **Latency sensitive** – CPU cache misses
  - **Bandwidth sensitive** – display out
  - **Best effort** - ethernet
  - Typically service classes have some traffic limit. Why?
- **Service guarantees** – achievable bandwidth/latency per service class
- **Fairness** – traffic from the same class should be prioritized equally
- How to implement?
  - Allocator/arbiter prioritization
  - Resource isolation

Intel Sandy Bridge



Scalable Ring On-die Interconnect

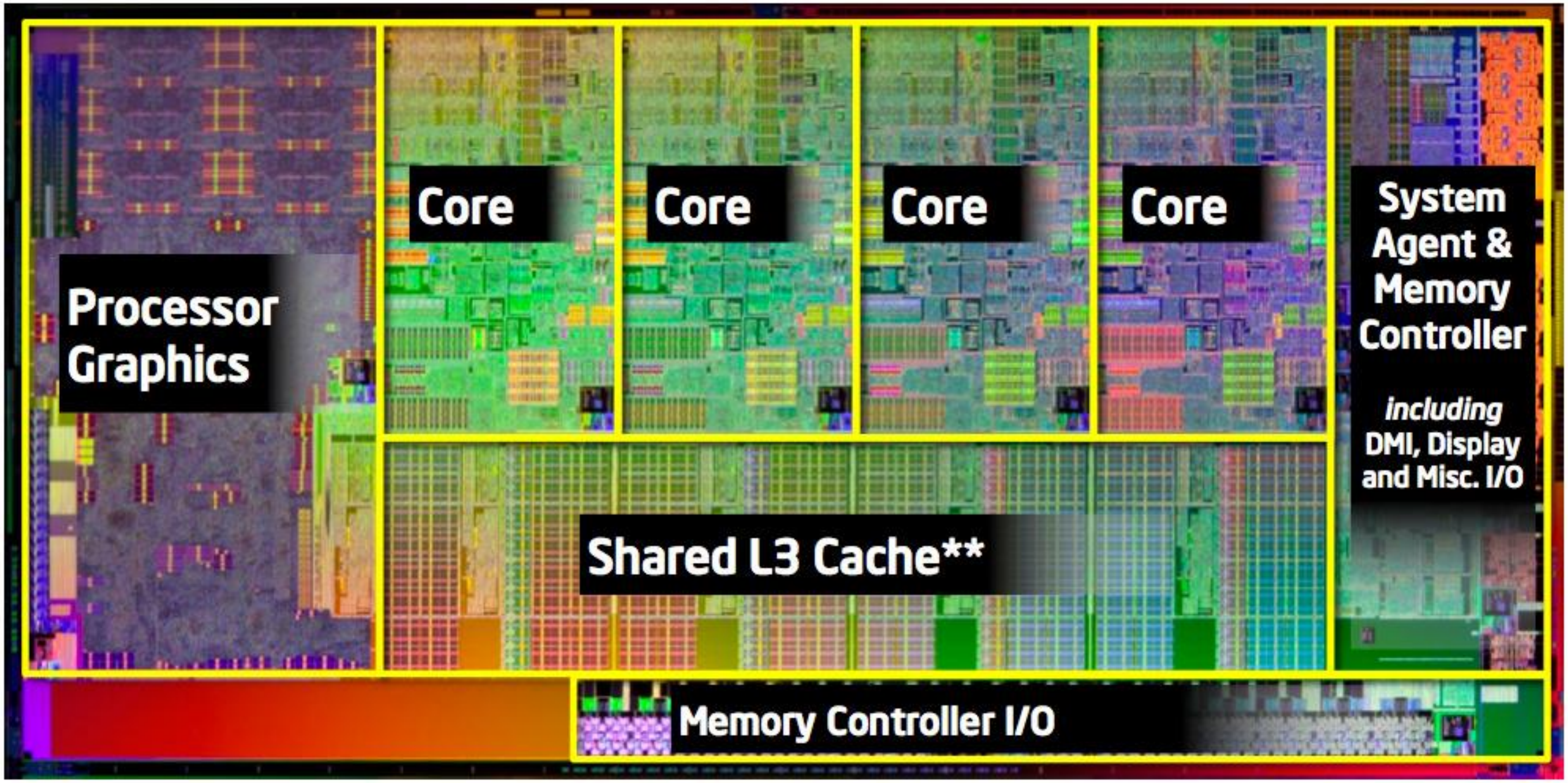
- **Ring-based** interconnect between Cores, Graphics, Last Level Cache (LLC) and System Agent domain
- Composed of **4 rings**
  - 32 Byte *Data* ring, *Request* ring, *Acknowledge* ring and *Snoop* ring
  - Fully pipelined at core frequency/voltage: bandwidth, latency and power scale with cores
- Massive ring **wire routing** runs over the LLC with no area impact
- Access on ring always picks the **shortest path** – minimize latency
- **Distributed arbitration**, sophisticated ring protocol to handle coherency, ordering, and core interface
- **Scalable to servers** with large number of processors



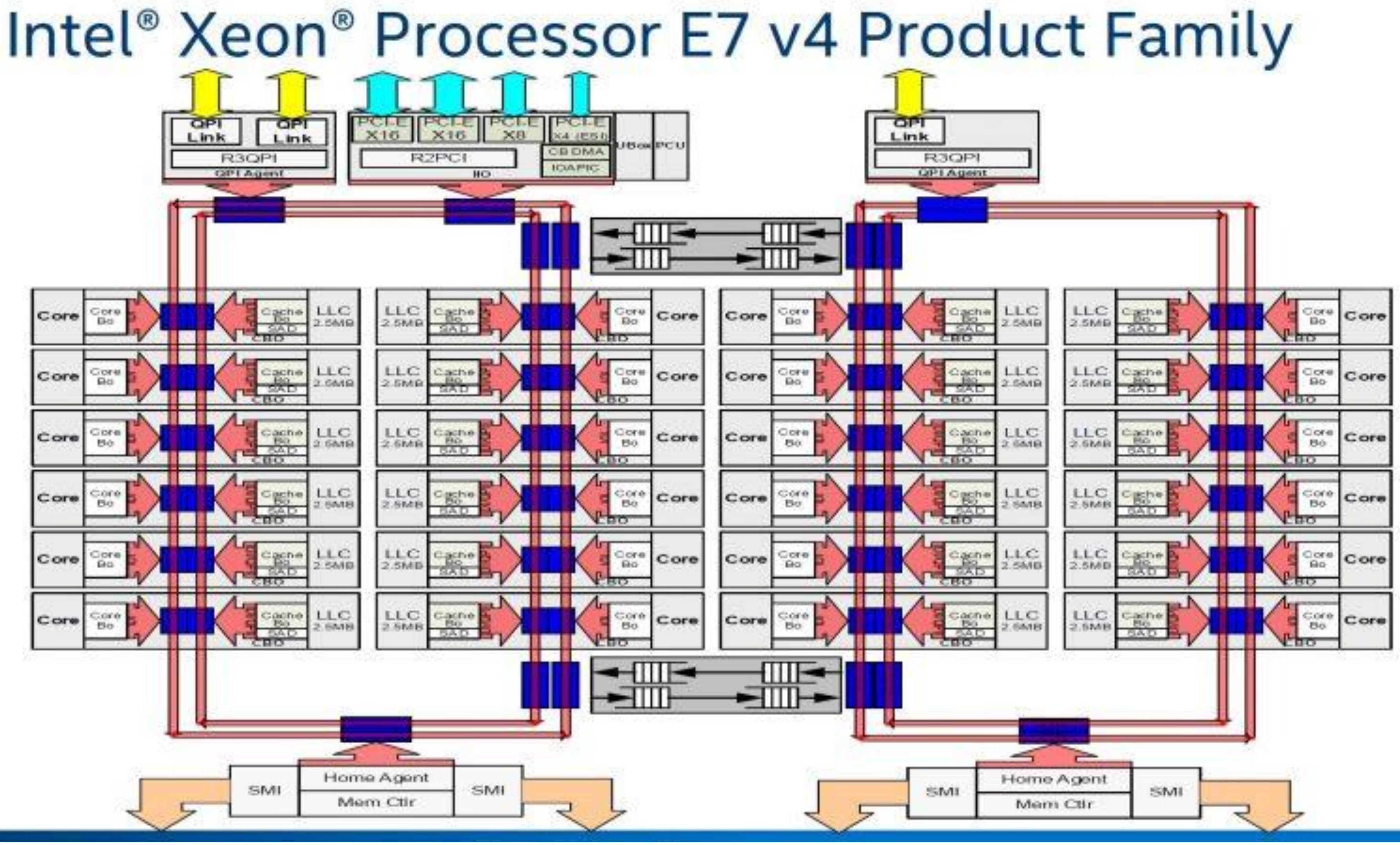
High Bandwidth, Low Latency, Modular

IDF2010

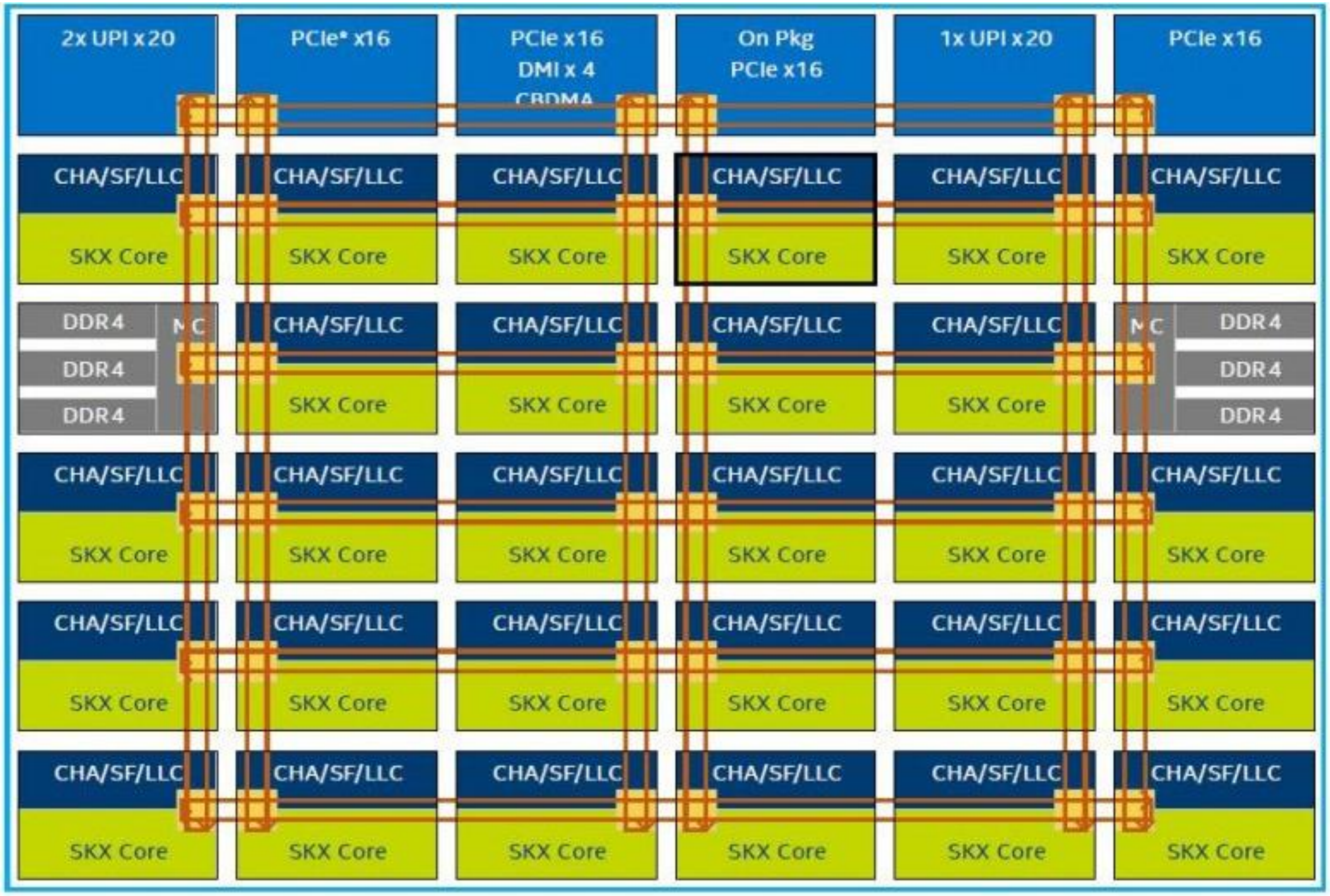
Intel Sandy Bridge



Intel Broadwell



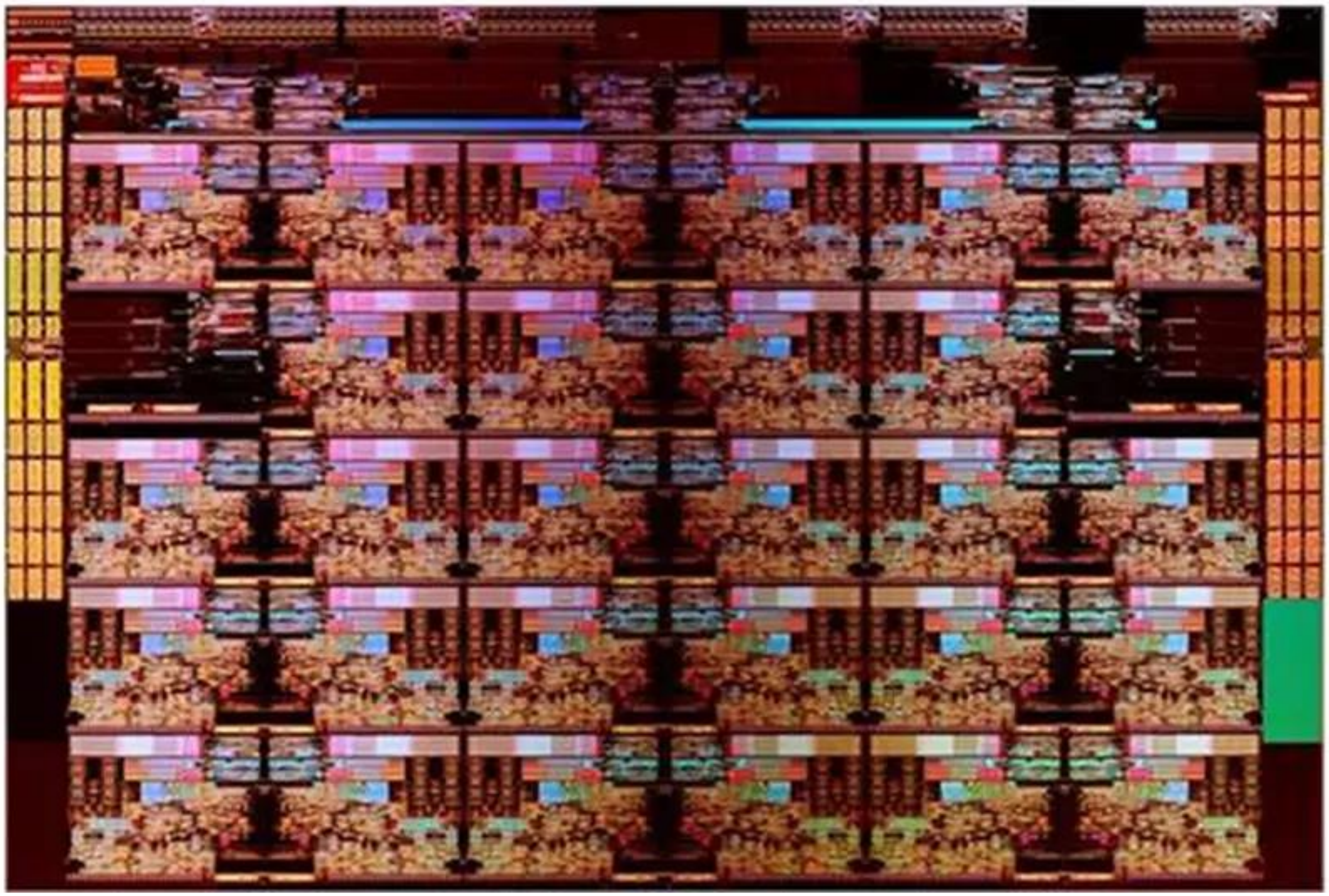
Intel Skylake



CHA – Caching and Home Agent ; SF – Snoop Filter; LLC – Last Level Cache;
SKX Core – Skylake Server Core; UPI – Intel® UltraPath Interconnect

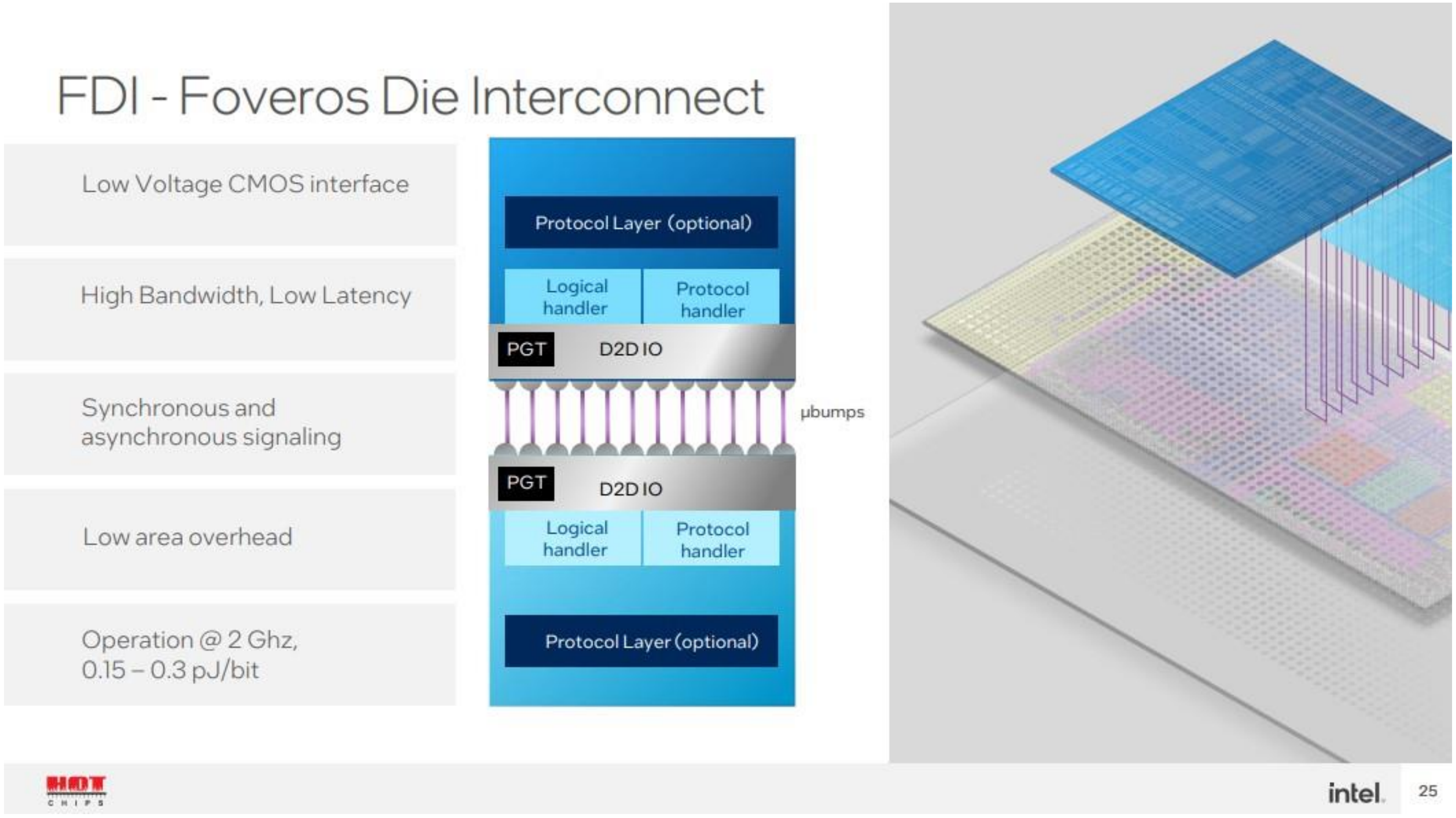
- 256b/cycle per channel (half a cache line)
- Dimension-ordered routing (First X, then Y)

Intel Skylake



Intel Meteor Lake

- 3D integration
- Separate SoC/GPU/CPU/IO Tiles
- “Base Tile” die implements routing



FDI - Foveros Die Interconnect

- Low Voltage CMOS interface
- High Bandwidth, Low Latency
- Synchronous and asynchronous signaling
- Low area overhead
- Operation @ 2 Ghz, 0.15 – 0.3 pJ/bit

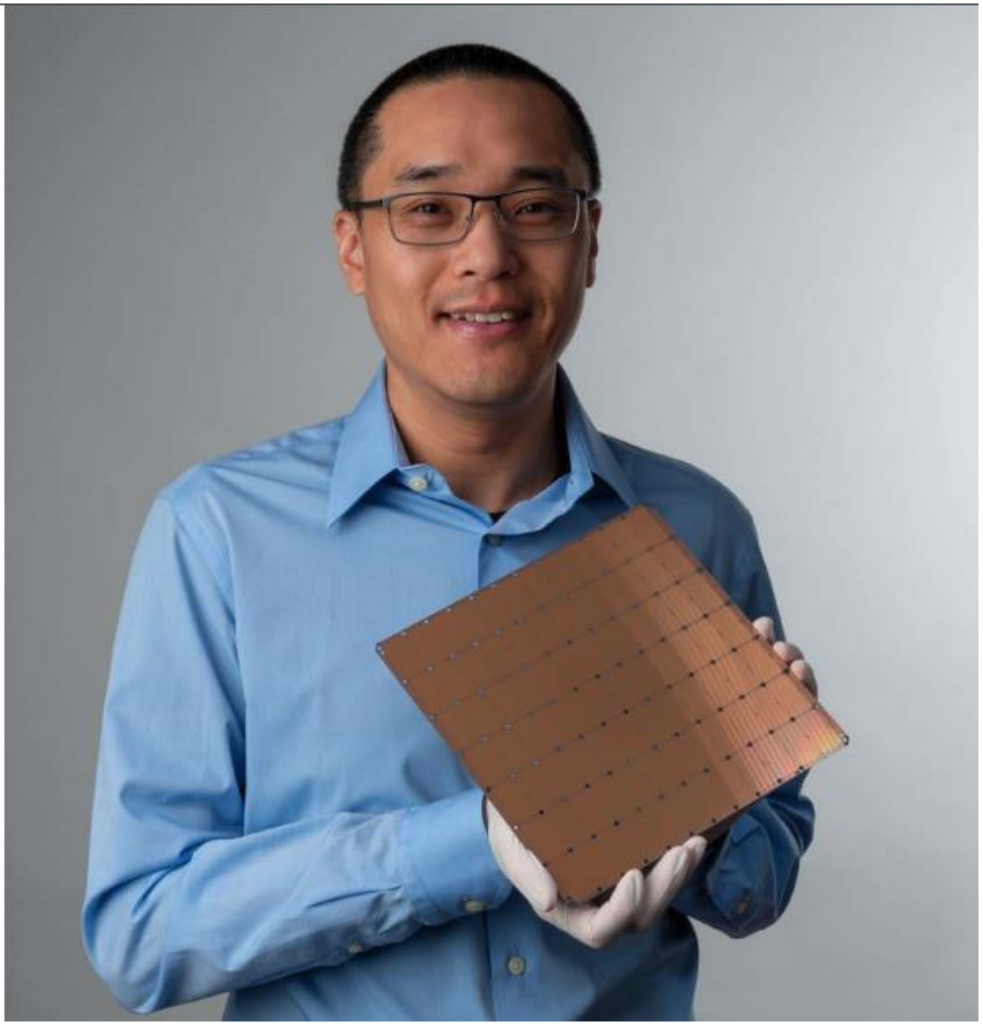
Cerebras WSE

- 2D mesh across an entire wafer

Largest Chip Ever Built

- 46,225 mm^2 silicon
- 1.2 trillion transistors
- 400,000 AI optimized cores
- 18 Gigabytes of On-chip Memory
- 9 PByte/s memory bandwidth
- 100 Pbit/s fabric bandwidth
- TSMC 16nm process

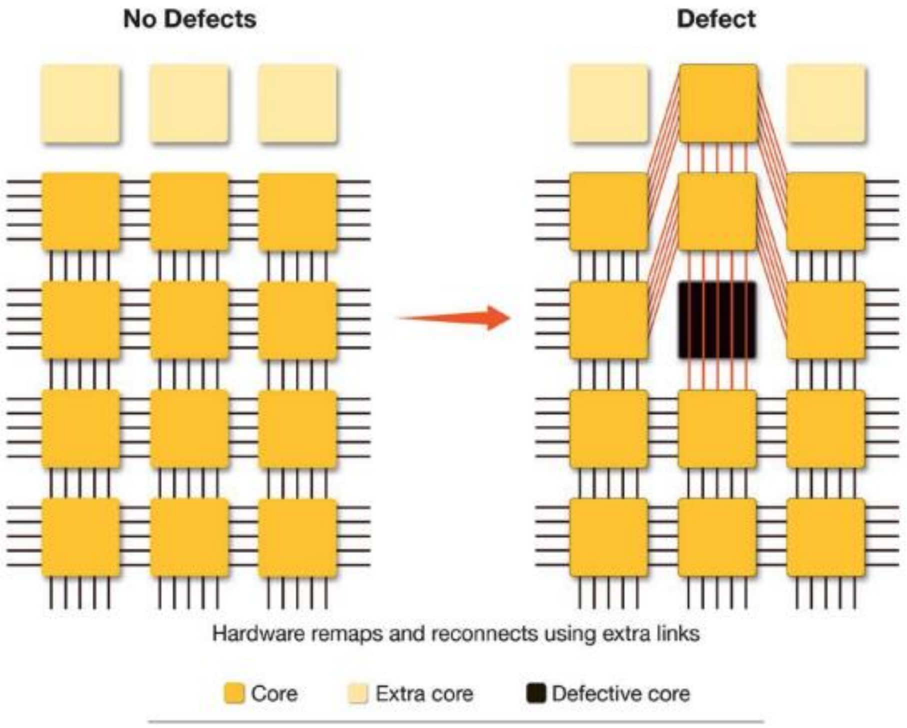
cerebras



Cerebras WSE

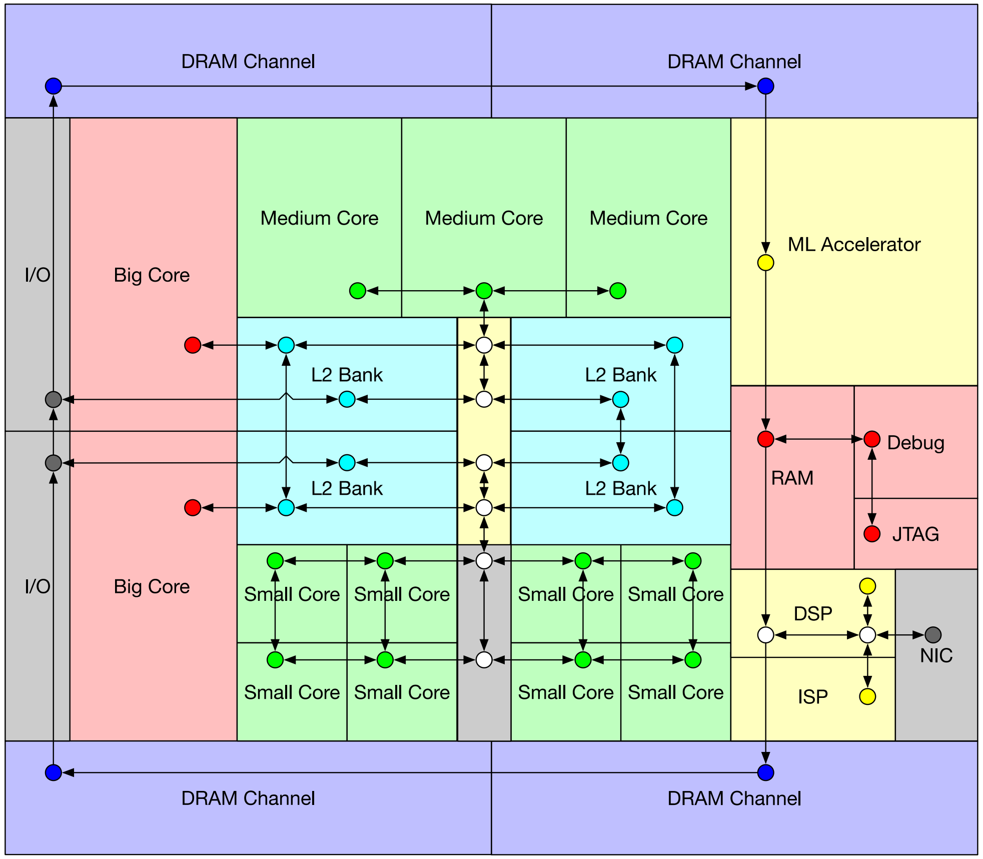
Redundancy is Your Friend

- Uniform small core architecture enables redundancy to address yield at very low cost
- Design includes redundant cores and redundant fabric links
- Redundant cores replace defective cores
- Extra links reconnect fabric to restore logical 2D mesh



Generating Network-on-Chips

- Constellation – a network-on-chip generator
- Generate realistic interconnects for modern SoCs
- Configurable routing/topology/micro-architecture
- If interested, email me – jzh@berkeley.edu



Acknowledgements

- Most diagrams are from the Dally and Towles textbook: *Principles and Practices of Interconnection Networks*
- Die shots, diagrams, slides are from HotChips presentations and AnandTech