Computer Architecture
Lecture 20: Interconnects

Prof. Onur Mutlu
ETH Zürich

Fall 2022

2 December 2022

Summary of Last Few Lectures

- Multiprocessing Fundamentals
- Memory Ordering (Consistency)
- Cache Coherence

Recall: Two Cache Coherence Methods

- How do we ensure that the proper caches are updated?

- **Snoopy Bus** [Goodman ISCA 1983, Papamarcos+ ISCA 1984]
  - Bus-based, single point of serialization *for all memory requests*
  - Processors observe other processors' actions
    - E.g.: P1 makes "read-exclusive" request for A on bus, P0 sees this and invalidates its own copy of A

- **Directory** [Censier and Feautrier, IEEE ToC 1978]
  - Single point of serialization *per block*, distributed among nodes
  - Processors make explicit requests for blocks
  - Directory tracks which caches have each block
  - Directory coordinates invalidation and updates
    - E.g.: P1 asks directory for exclusive copy, directory asks P0 to invalidate, waits for ACK, then responds to P1

Recall: Snoopy Cache vs. Directory Coherence

- **Snoopy Cache**
  - + Miss latency (critical path) is short: request → bus transaction to mem.
  - + Global serialization is easy: bus provides this already (arbitration)
  - + Simple: can adapt bus-based uniprocessors easily
  - – Relies on broadcast messages to be seen by all caches (in same order):
    - → single point of serialization (bus): *not scalable*
    - → *need a virtual bus (or a totally-ordered interconnect)*

- **Directory**
  - – Adds indirection to miss latency (critical path): request → dir. → mem.
  - – Requires extra storage space to track sharer sets
    - Can be approximate (false positives are OK for correctness)
  - – Protocols and race conditions are more complex (for high-performance)
  - + Does not require broadcast to all caches
  - + Exactly as scalable as interconnect and directory storage *(much more scalable than bus)*

Interconnection Networks

Readings

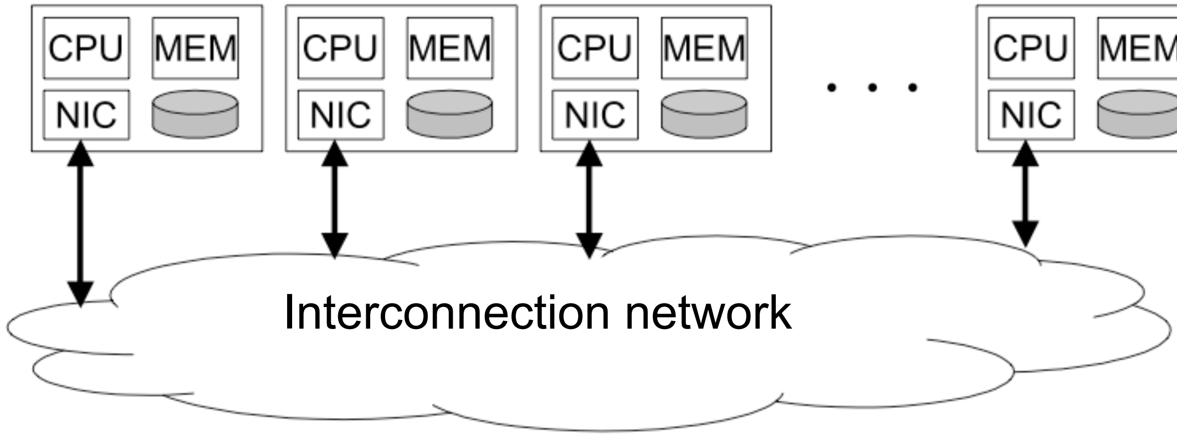
- Required
  - Moscibroda & Mutlu, "A Case for Bufferless Routing in On-Chip Networks," ISCA 2009.
  - Das et al., "Aergia: Exploiting Packet Latency Slack in On-Chip Networks," ISCA 2010.
- Recommended
  - Das et al., "Application-Aware Prioritization Mechanisms for On-Chip Networks," MICRO 2009.
  - Dally & Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," DAC 2001.
  - Janak H. Patel,"Processor-Memory Interconnections for Multiprocessors," ISCA 1979.
  - Gottlieb et al. "The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer," IEEE Trans. On Comp., 1983.

Interconnection Network Basics

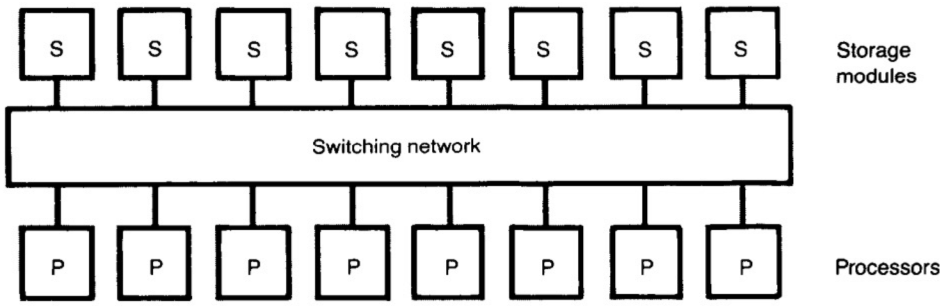
Where Is Interconnect Used?

- To connect & communicate between components

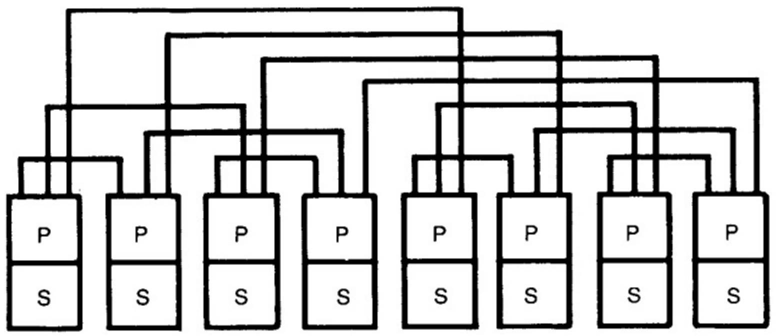
- Many examples
  - Processors and processors
  - Processors and memories (banks)
  - Processors and caches (banks)
  - Caches and caches
  - I/O devices



Interconnects Enable Communication



(a) Most multiprocessors are structured with a switching network, either a crossbar connection of buses or a multistage routing network, between the processors and storage. The switching network introduces a latency in the communication between processors and storage, and does not scale well to large sizes. Communication between processes running concurrently in different processors occurs through shared variables and common access to one large address space.



(b) Message-passing multicomputer systems retain a physically close and fast connection between processors and their associated storage. The concurrent computers (nodes) can send messages through a network of communication channels. The network shown here is a three-dimensional cube, which is a small version of the communication plan used in six dimensions in the 64-node Cosmic Cube.

NOTE: Actual machines need not follow one model or the other absolutely: Various hybrids are possible.

FIGURE 2. A Comparison of Shared-Storage Multiprocessors and Message-Passing Machines

Seitz, "The Cosmic Cube," CACM 1985.

Why Is It Important?

- Affects the scalability and cost of the system
  - How large of a system can you build?
  - How easily can you add more processors?

- Affects performance and energy efficiency
  - How fast can processors, caches, and memory communicate?
  - How long are the latencies to memory?
  - How much energy is spent on communication?

- Affects reliability and security
  - Can you guarantee messages are delivered or your protocol works?

Many Parameters & Goals in a Network

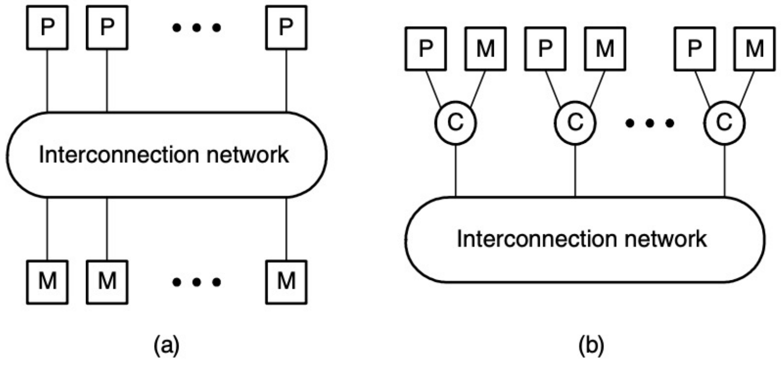


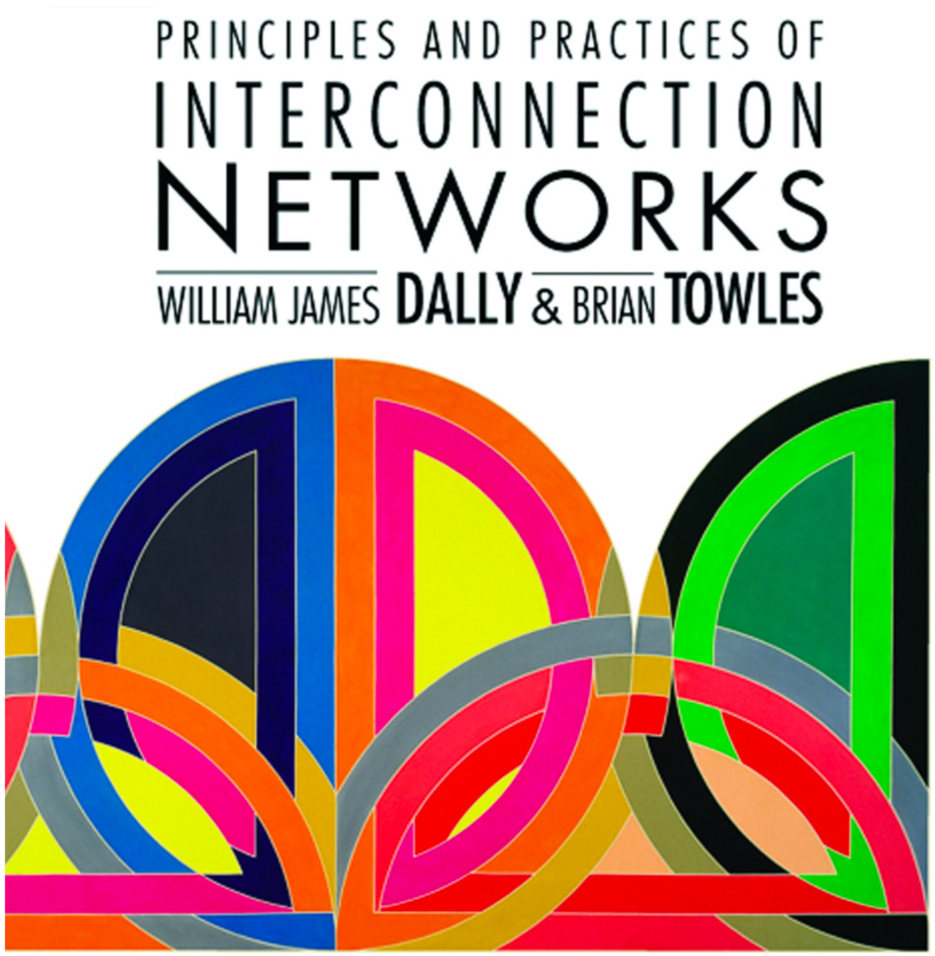
Figure 1.2 Use of an interconnection network to connect processor and memory. (a) *Dance-hall* architecture with separate processor (P) and memory (M) ports. (b) Integrated-node architecture with combined processor and memory ports and local access to one memory bank.

Table 1.1 Parameters of processor-memory interconnection networks.

Parameter	*Value*
Processor ports	1–2,048
Memory ports	0–4,096
Peak bandwidth	8 Gbytes/s
Average bandwidth	400 Mbytes/s
Message latency	100 ns
Message size	64 or 576 bits
Traffic patterns	arbitrary
Quality of service	none
Reliability	no message loss
Availability	0.999 to 0.99999

Dally & Towles, "Principles and Practices of Interconnection Networks," 2004.

A Recommended Book


PRINCIPLES AND PRACTICES OF
INTERCONNECTION NETWORKS
WILLIAM JAMES DALLY & BRIAN TOWLES

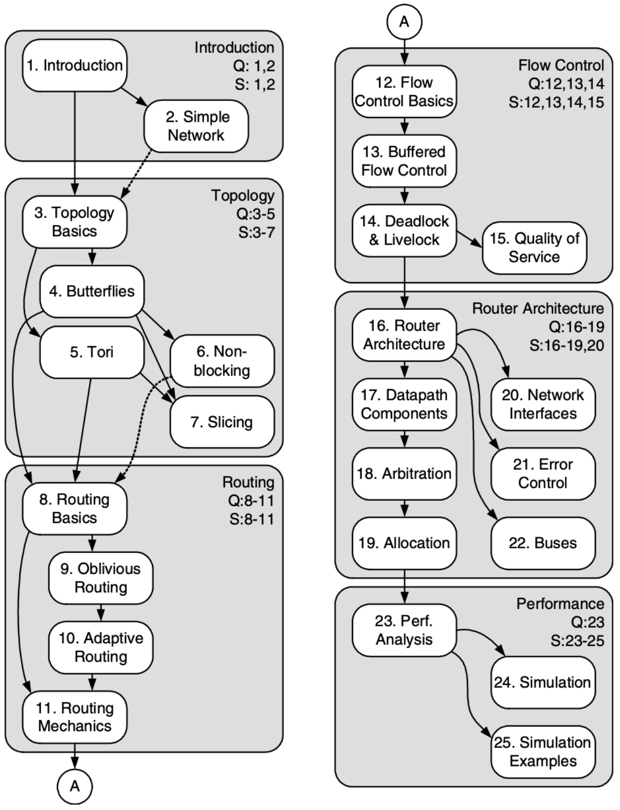


Figure 1 Outline of this book showing dependencies between chapters. Major sections are denoted as shaded areas. Chapters that should be covered in any course on the subject are placed along the left side of the shaded areas. Optional chapters are placed to the right. Dependences are indicated by arrows. A solid arrow implies that the chapter at the tail of the arrow must be understood to understand the chapter at the head of the arrow. A dotted arrow indicates that it is helpful, but not required, to understand the chapter at the tail of the arrow before the chapter at the head. The notation in each shaded area recommends which chapters to cover in a quarter course (Q) and a semester course (S).

Dally & Towles, "Principles and Practices of Interconnection Networks," 2004.

Another Example: Clock Distribution Network

- Problem: Clock signal arrives non-uniformly late to different parts of a chip, causing potential timing issues
- Solution: Design the interconnect to equalize the arrival time of the clock signal to all parts of a chip
- This specialized interconnect communicates the "clock" signal

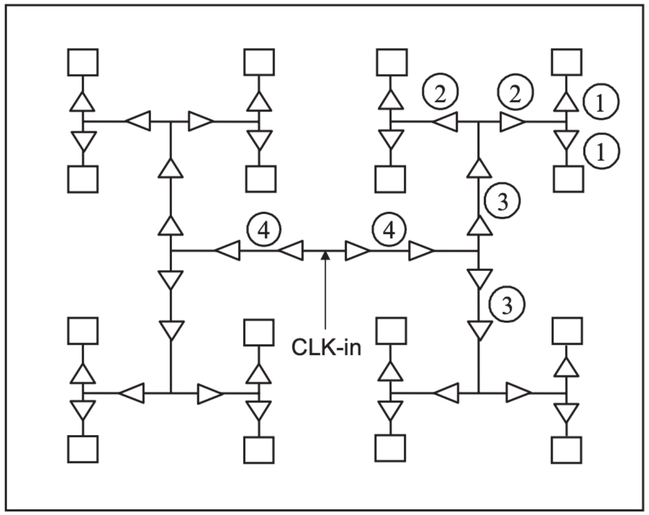


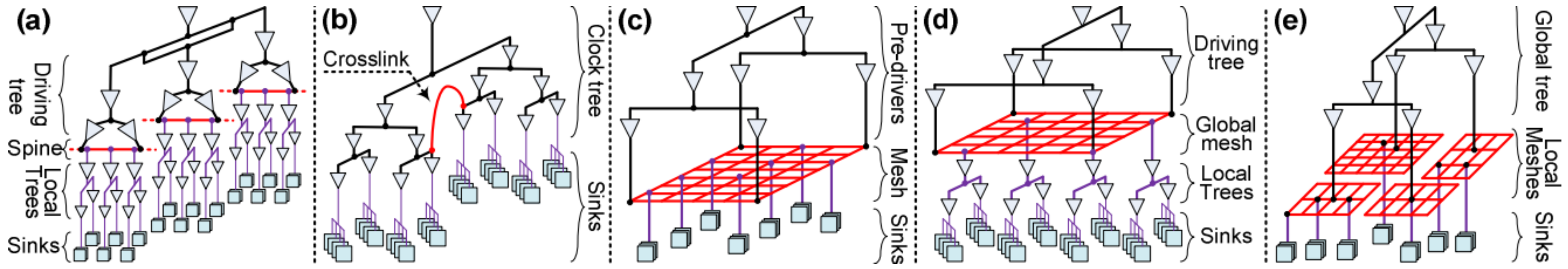
Fig. 7. An H-tree clock distribution network. The index of each level is indicated with the circled numbers. The squares represent the fixed loads at the leaves.

Image Source: Tawfik & Kursun, "Clock Distribution Networks with Gradual Signal Transition Time Relaxation for Reduced Power Consumption," JCSS 2008.

Recall: Clock Skew: Summary

- **Clock Skew** effectively **increases** both $\mathbf{t_{setup}}$ and $\mathbf{t_{hold}}$
  - Increased **sequencing overhead**
  - i.e., less useful work done per cycle

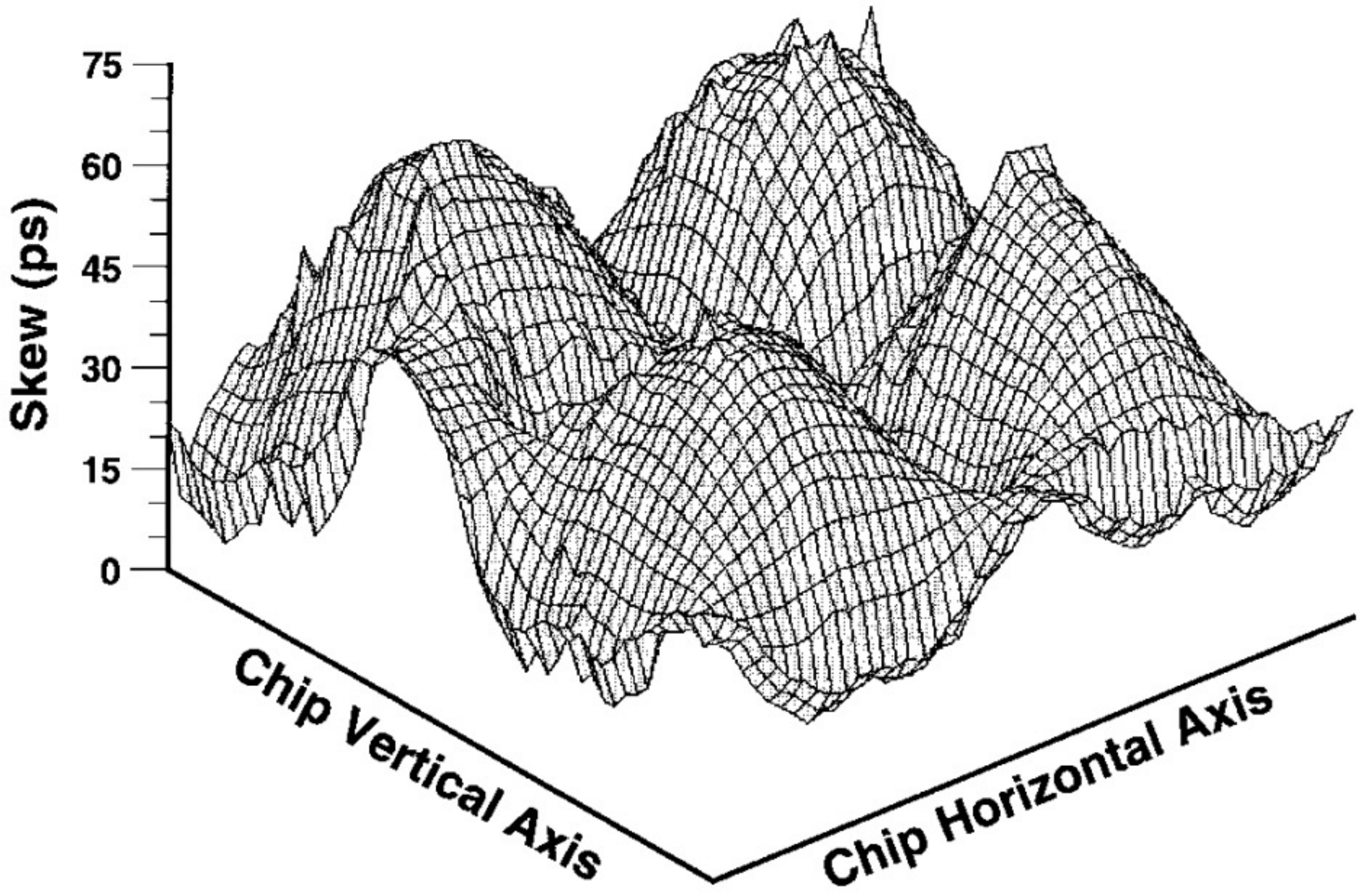
- Designers must keep clock skew to a **minimum**
  - Requires intelligent **"clock network"** across a chip
  - **Goal: clock** arrives at all locations at roughly the **same time**



Source: Abdelhadi, Ameer, et al. "Timing-driven variation-aware nonuniform clock mesh synthesis." GLSVLSI'10.

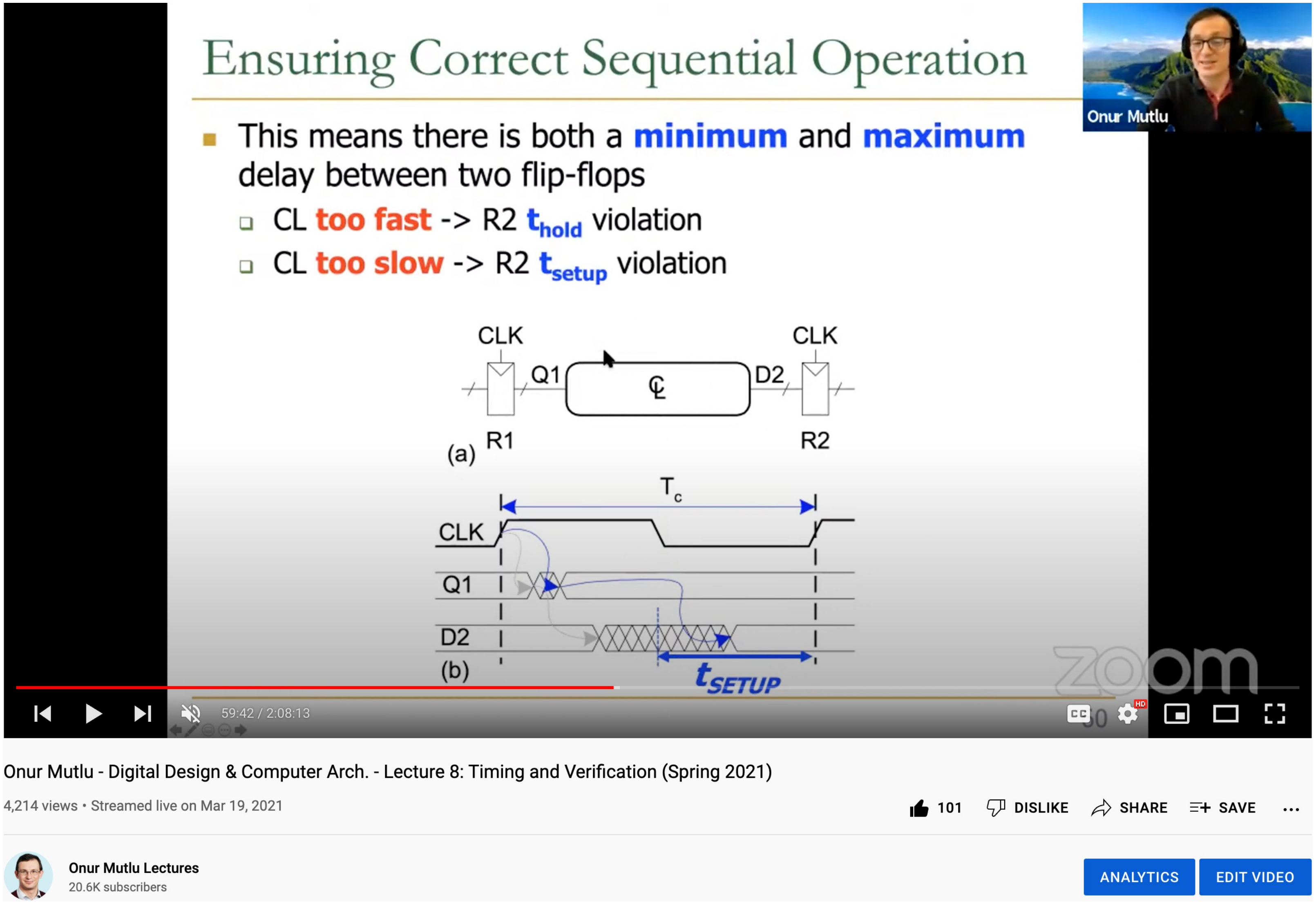
Recall: Clock Skew Example

- Example of the **Alpha 21264** clock skew spatial distribution



P. E. Gronowski+, "High-performance Microprocessor Design," JSSC'98.

Recall: Timing & Verification



Ensuring Correct Sequential Operation

- This means there is both a **minimum** and **maximum** delay between two flip-flops
  - CL **too fast** -> R2 t_{hold} violation
  - CL **too slow** -> R2 t_{setup} violation

Onur Mutlu - Digital Design & Computer Arch. - Lecture 8: Timing and Verification (Spring 2021)

4,214 views • Streamed live on Mar 19, 2021

https://www.youtube.com/watch?v=4AcwBBwiNlQ&list=PL5Q2soXY2Zi_uej3aY39YB5pfW4SJ7LlN&index=8

Interconnection Network Basics

- Topology
  - Specifies the way switches are wired
  - Affects routing, reliability, throughput, latency, building ease

- Routing (algorithm)
  - How does a message get from source to destination
  - Static or adaptive

- Buffering and Flow Control
  - What do we store within the routers & links?
    - Entire packets, parts of packets, etc?
  - How do we throttle during oversubscription?
  - Tightly coupled with routing strategy

Terminology

- Network interface
  - Module that connects endpoints (e.g. processors) to network
  - Decouples computation/communication
- Link
  - Bundle of wires that carry a signal
- Switch/router
  - Connects fixed number of input channels to fixed number of output channels
- Channel
  - A single logical connection between routers/switches

More Terminology

- Node
  - A router/switch within a network
- Message
  - Unit of transfer for network’s clients (processors, memory)
- Packet
  - Unit of transfer for network
- Flit
  - Flow control digit
  - Unit of flow control within network

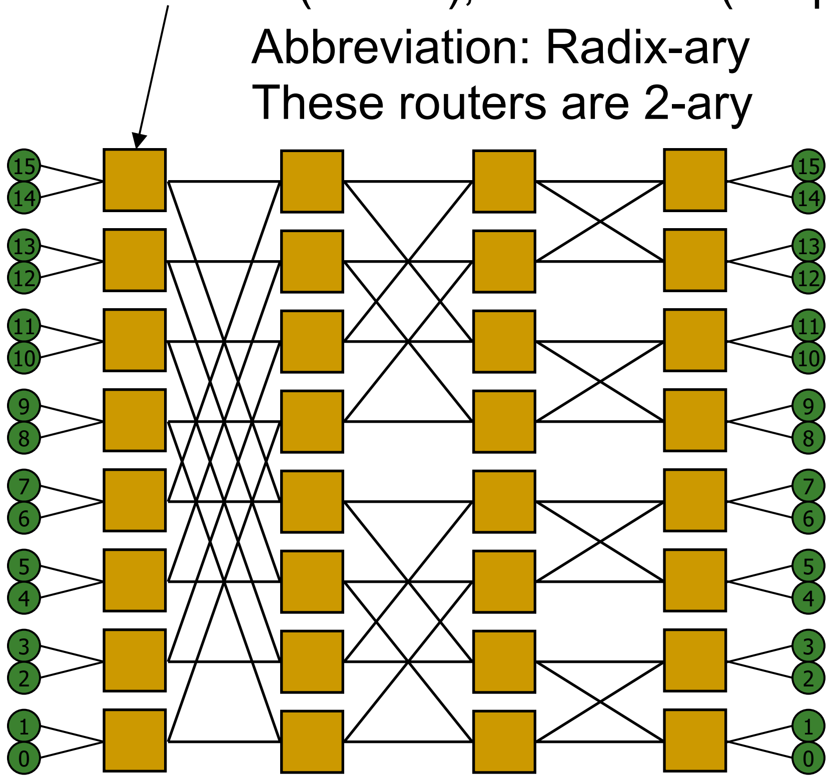
Some More Terminology

- Direct or Indirect Networks
  - Endpoints sit "inside" (direct) or "outside" (indirect) the network
  - E.g. mesh is direct; every node is both endpoint and switch

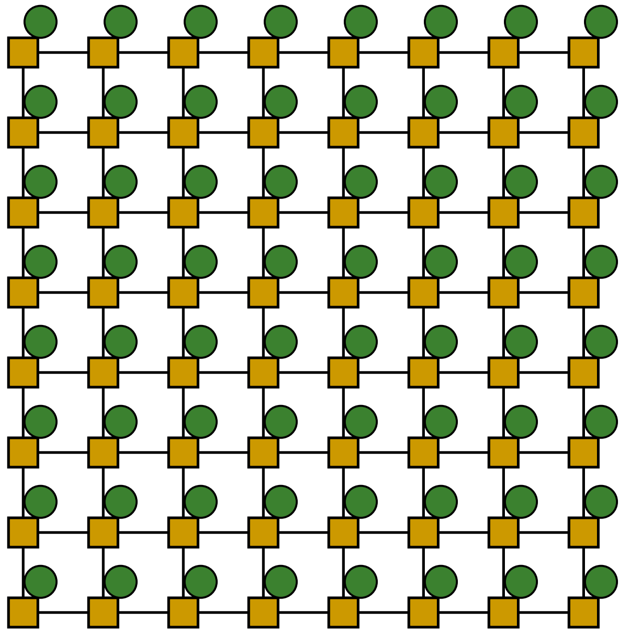
Router (switch), Radix of 2 (2 inputs, 2 outputs)

Abbreviation: Radix-ary

These routers are 2-ary



Indirect



Direct

Interconnection Network Topology

Properties of a Topology/Network

- Regular or Irregular
  - Regular if topology is a regular graph (e.g., ring, mesh).
- Routing Distance
  - number of links/hops along a route
- Diameter
  - maximum routing distance within the network
- Average Distance
  - Average number of hops across all valid routes

Properties of a Topology/Network

- Bisection Bandwidth
  - Often used to describe network performance
  - Cut network in half and sum bandwidth of links severed
    - (Min # channels spanning two halves) * (BW of each channel)
  - Meaningful only for recursive topologies
  - Can be misleading, because does not account for switch and routing efficiency (and certainly not execution time)

- Blocking vs. Non-Blocking
  - If connecting any permutation of sources & destinations is possible, network is <u>non-blocking</u>; otherwise network is <u>blocking</u>.
  - <u>Rearrangeable non-blocking</u>: Same as non-blocking but might require rearranging connections when switching from one permutation to another.

Topology

- Bus (simplest)
- Point-to-point connections (most costly)
- Crossbar
- Ring
- Tree
- Omega
- Hypercube
- Mesh
- Torus
- Butterfly
- …

Metrics to Evaluate Interconnect Topology

- Cost
- Latency (in hops, in nanoseconds)
- Contention

- Many others exist you should think about
  - Energy
  - Bandwidth
  - Overall system performance

Bus

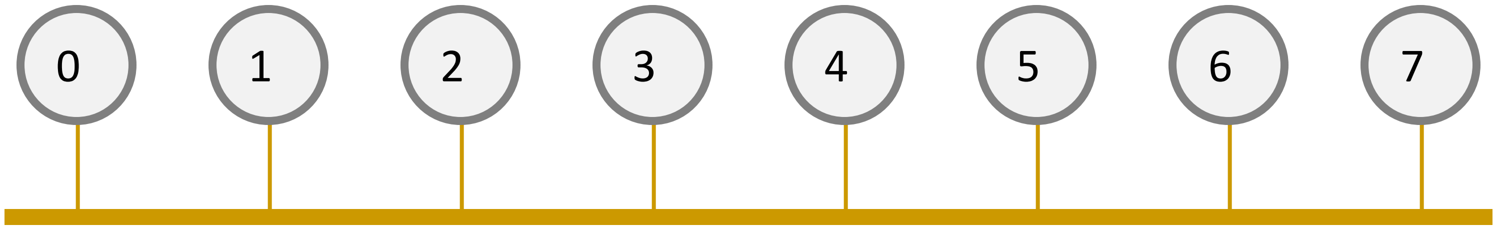
All nodes connected to a single link

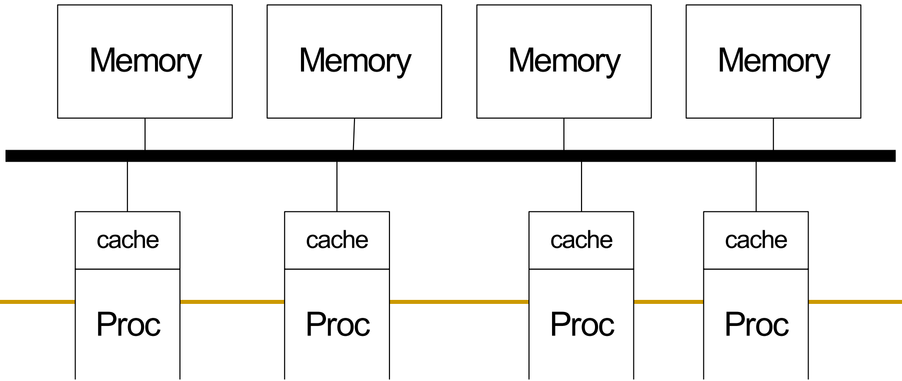
+ Simple + Cost effective for a small number of nodes

+ Easy to implement coherence (snooping and serialization)

- Not scalable to large number of nodes (limited bandwidth, electrical loading → reduced frequency)

- High contention → fast saturation



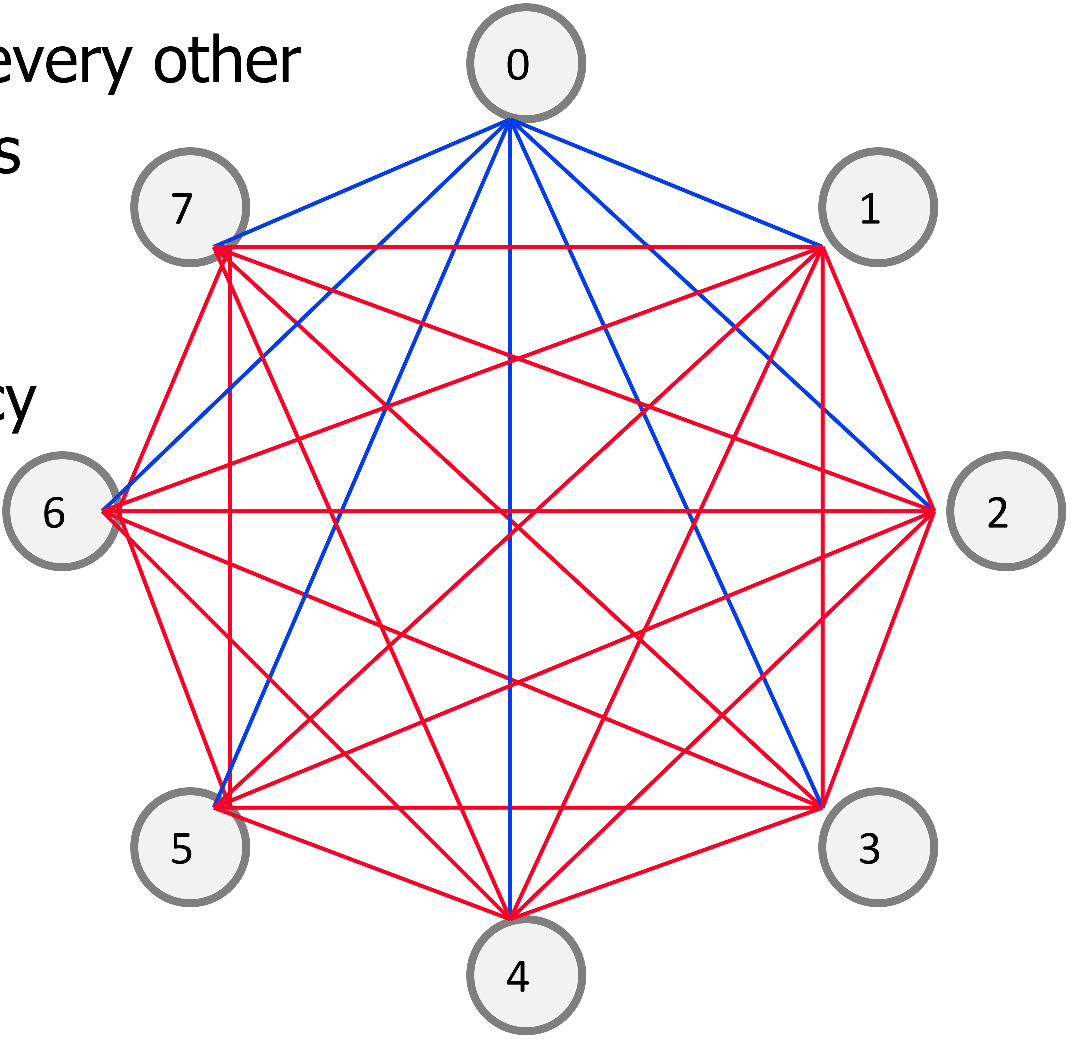


Point-to-Point

Every node connected to every other
with direct/isolated links

+ Lowest contention
+ Potentially lowest latency
+ Ideal, if cost is no issue

-- Highest cost
O(N) connections/ports
per node
O(N^2) links
-- Not scalable
-- How to lay out on chip?



Crossbar

- Every node connected to every other with a shared link for each destination
- Enables concurrent transfers to non-conflicting destinations
- Could be cost-effective for small number of nodes

+ Low latency and high throughput

- Expensive

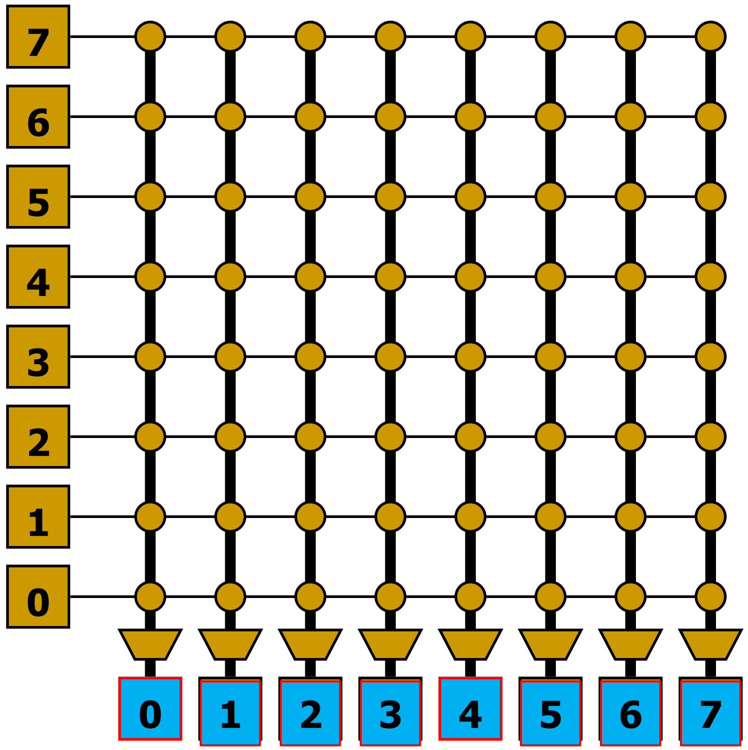
- Not scalable → O(N^2) cost

- Difficult to arbitrate as N increases

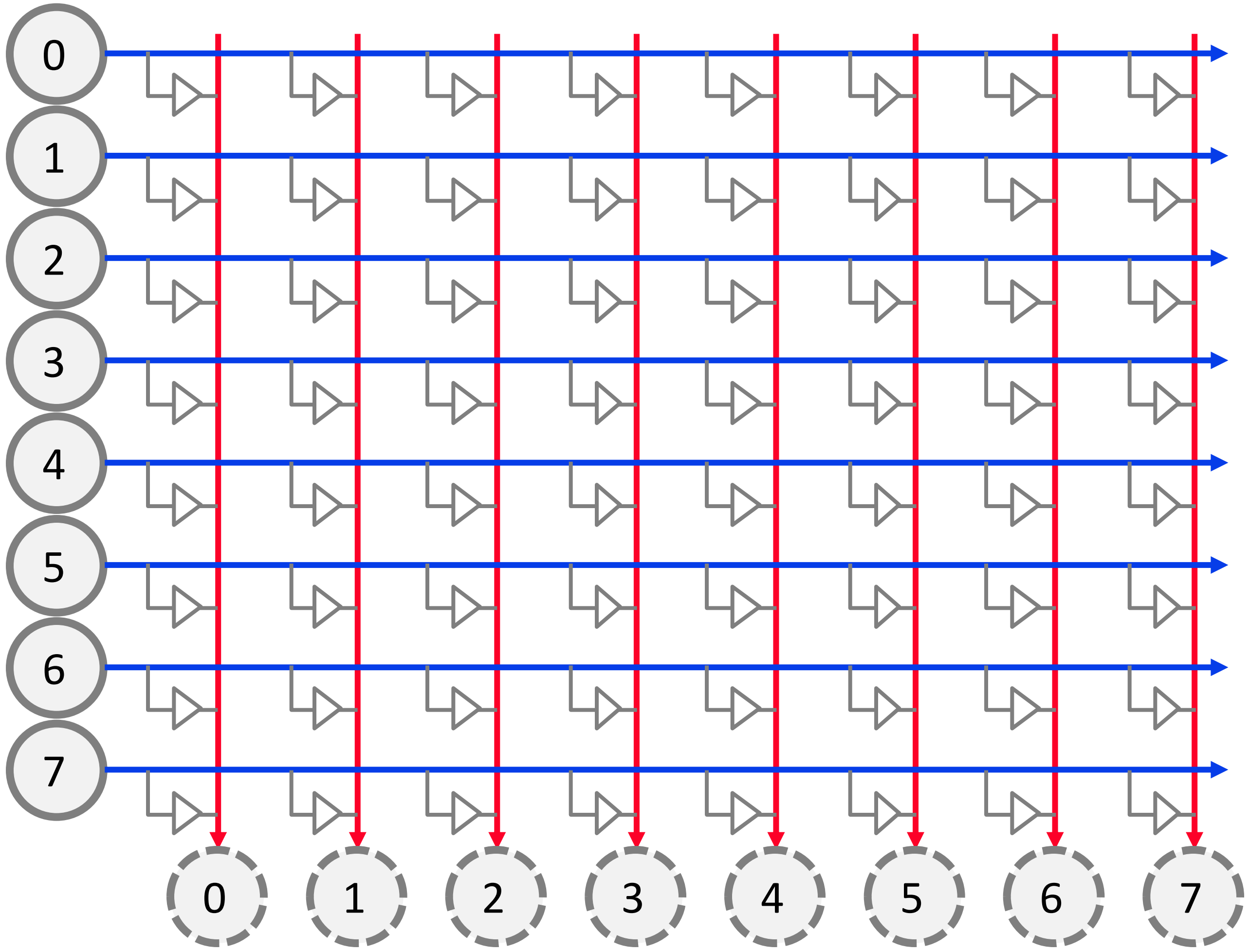
Used in core-to-cache-bank networks in

- IBM POWER5

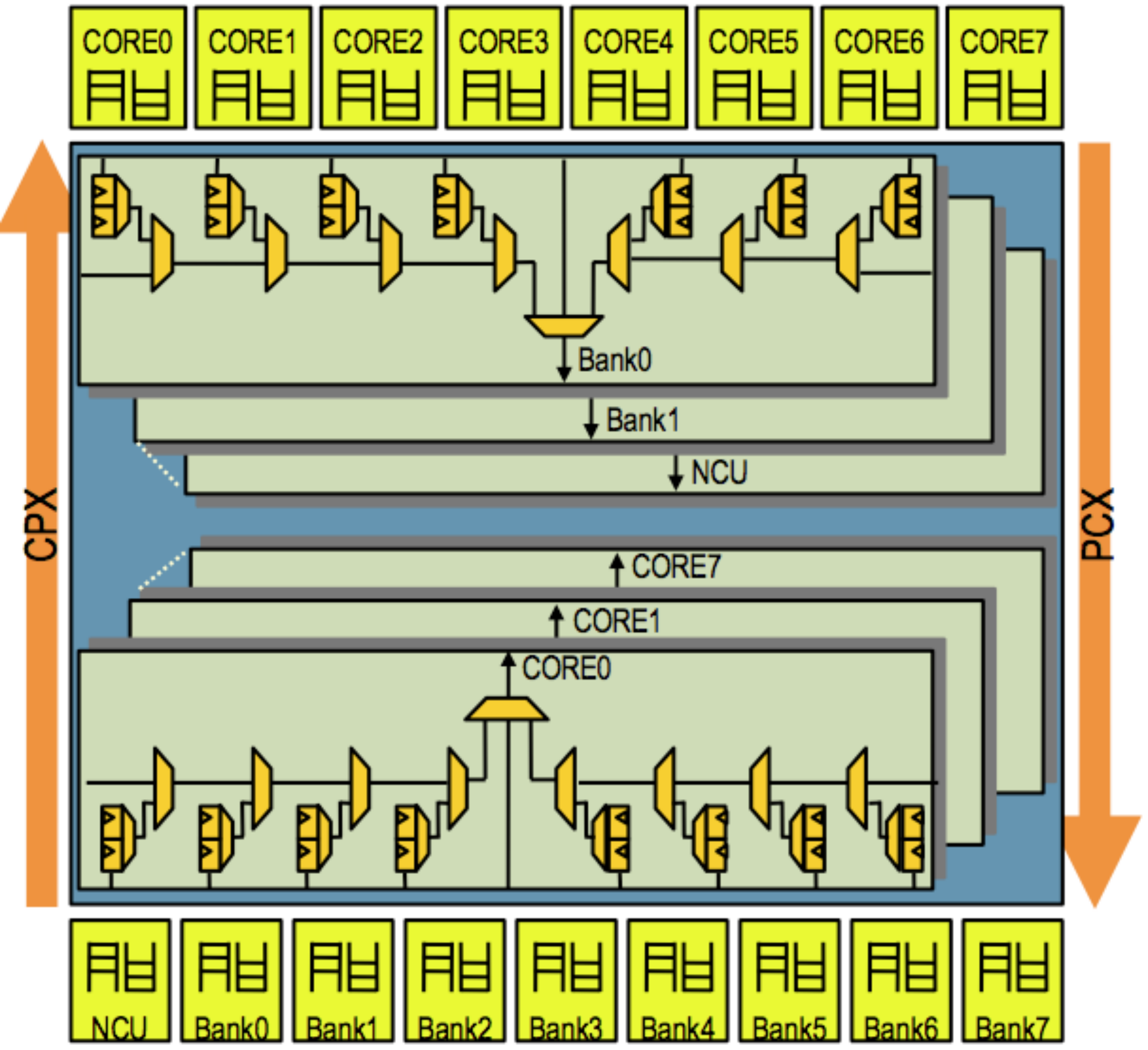
- Sun Niagara I/II



Another Crossbar Design



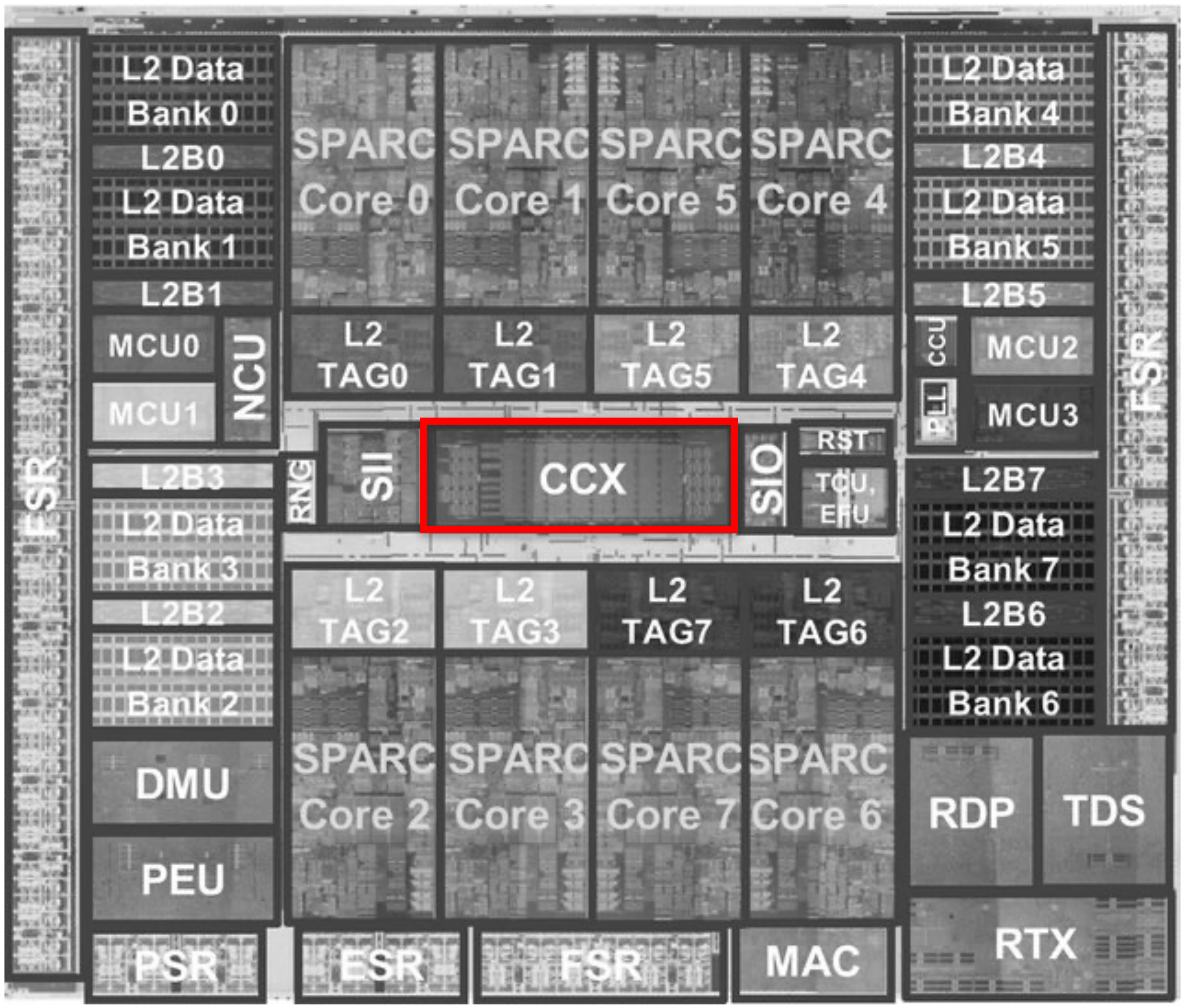
Sun UltraSPARC T2 Core-to-Cache Crossbar



- High bandwidth interface between 8 cores and 8 L2 banks & NCU
- 4-stage pipeline: req, arbitration, selection, transmission
- 2-deep queue for each requestor to hold data transfer request

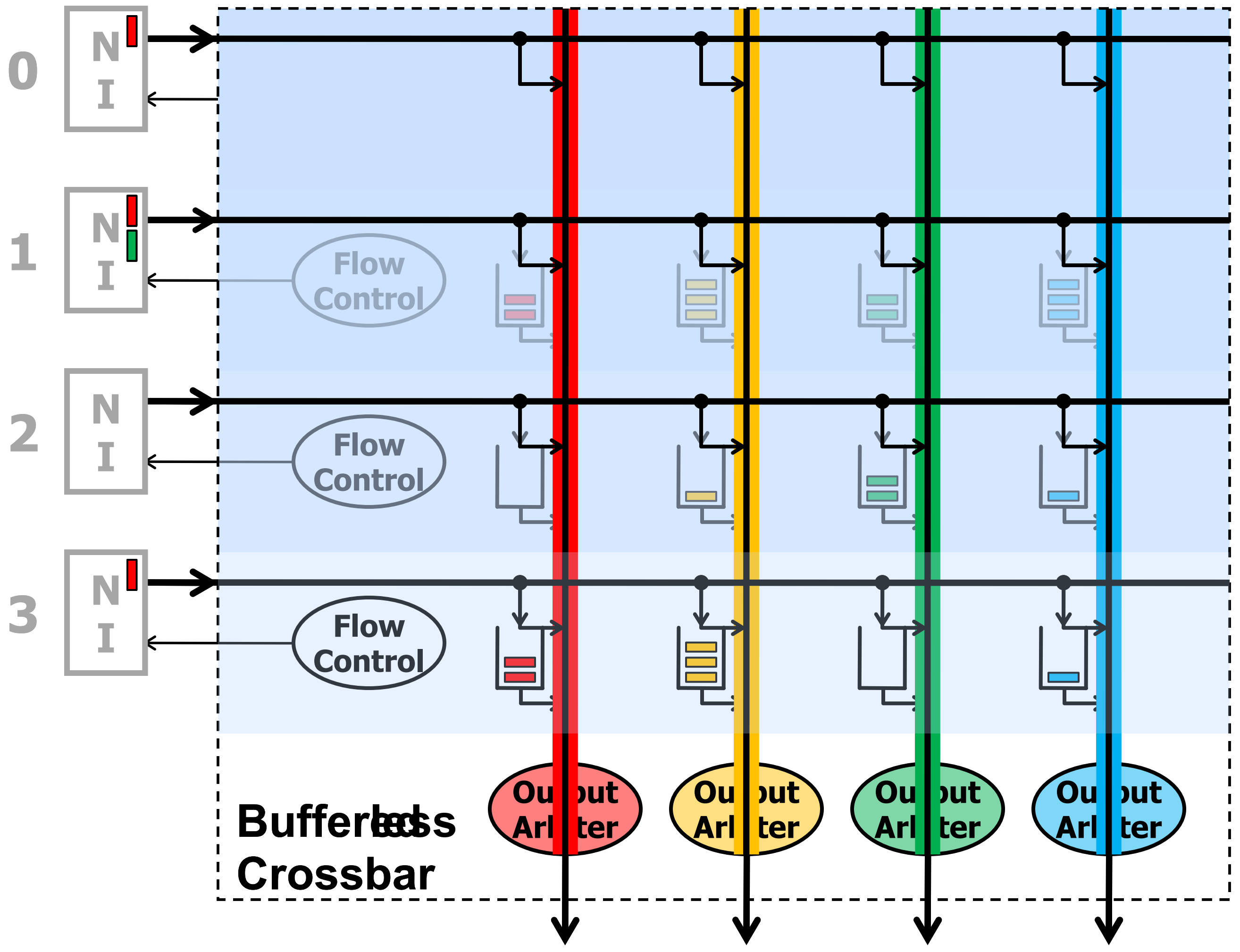
]

Sun UltraSPARC T2 Core-to-Cache Crossbar



Nawathe+, "Implementation of an 8-Core, 64-Thread, Power-Efficient SPARC Server on a Chip," JSSC 2008.

Bufferless and Buffered Crossbars



+ Simpler arbitration/ scheduling

+ Efficient support for variable-size packets

- Requires N^2 buffers

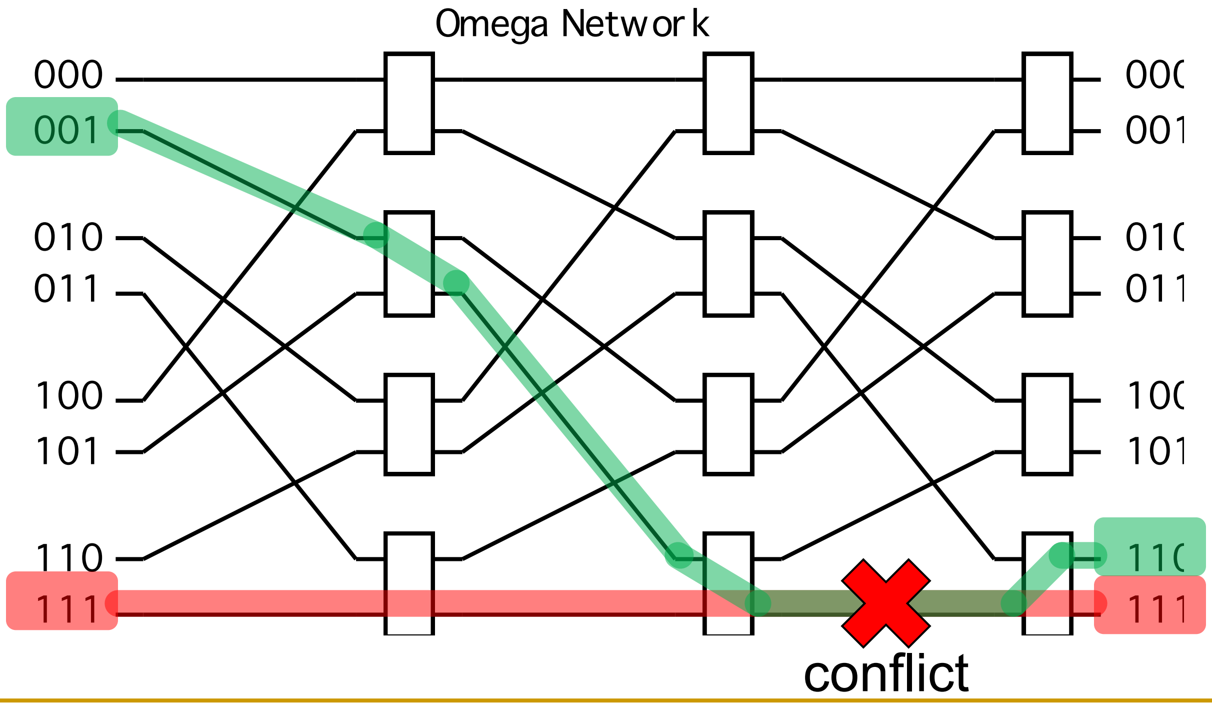
Can We Get Lower Cost than A Crossbar?

- Yet still have low contention compared to a bus?

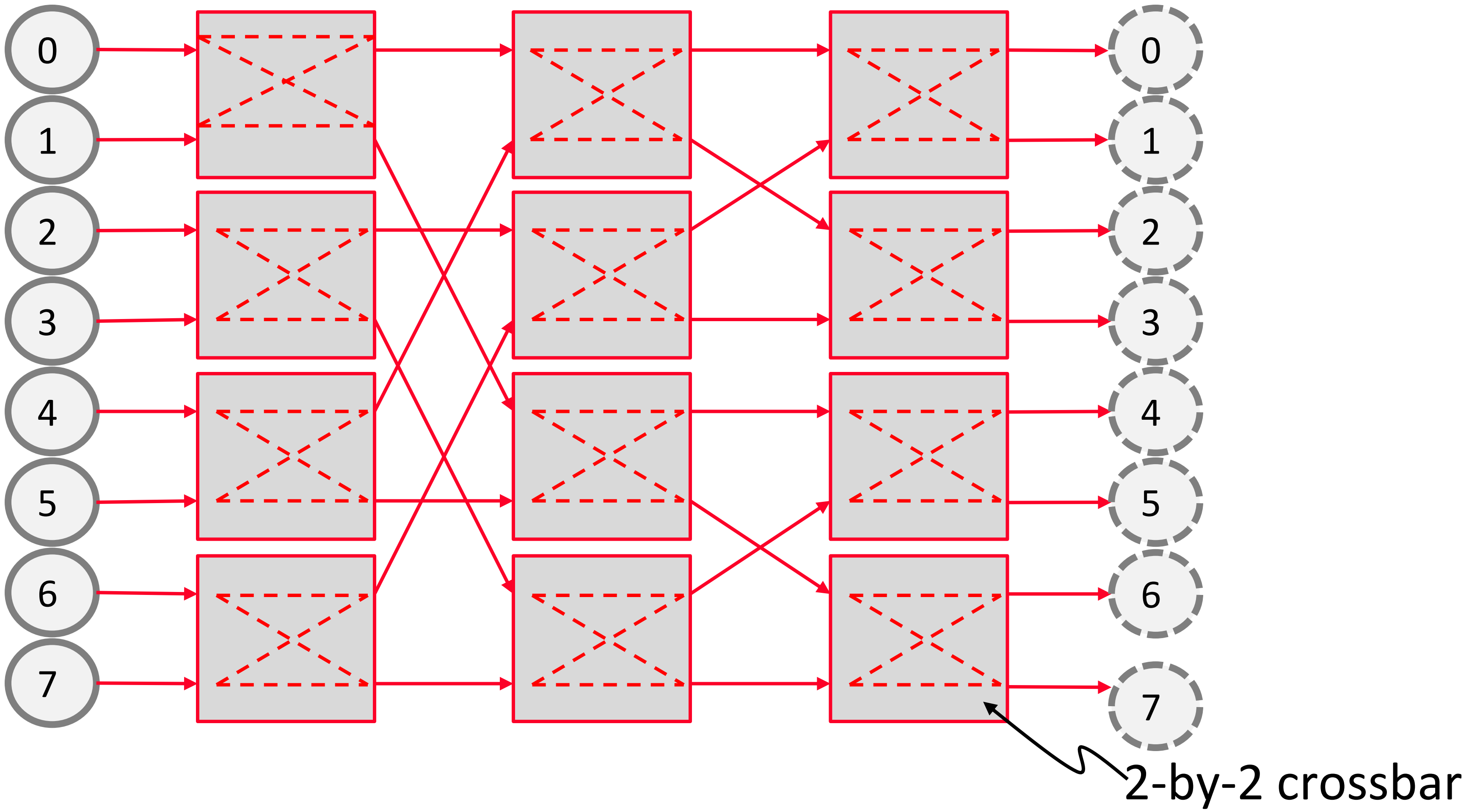
- Idea: Multistage networks

Multistage Logarithmic Networks

- Idea: Indirect networks with multiple layers of switches between terminals/nodes
- Cost: O(NlogN), Latency: O(logN)
- Many variations (Omega, Butterfly, Benes, Banyan, …)
- Omega Network:

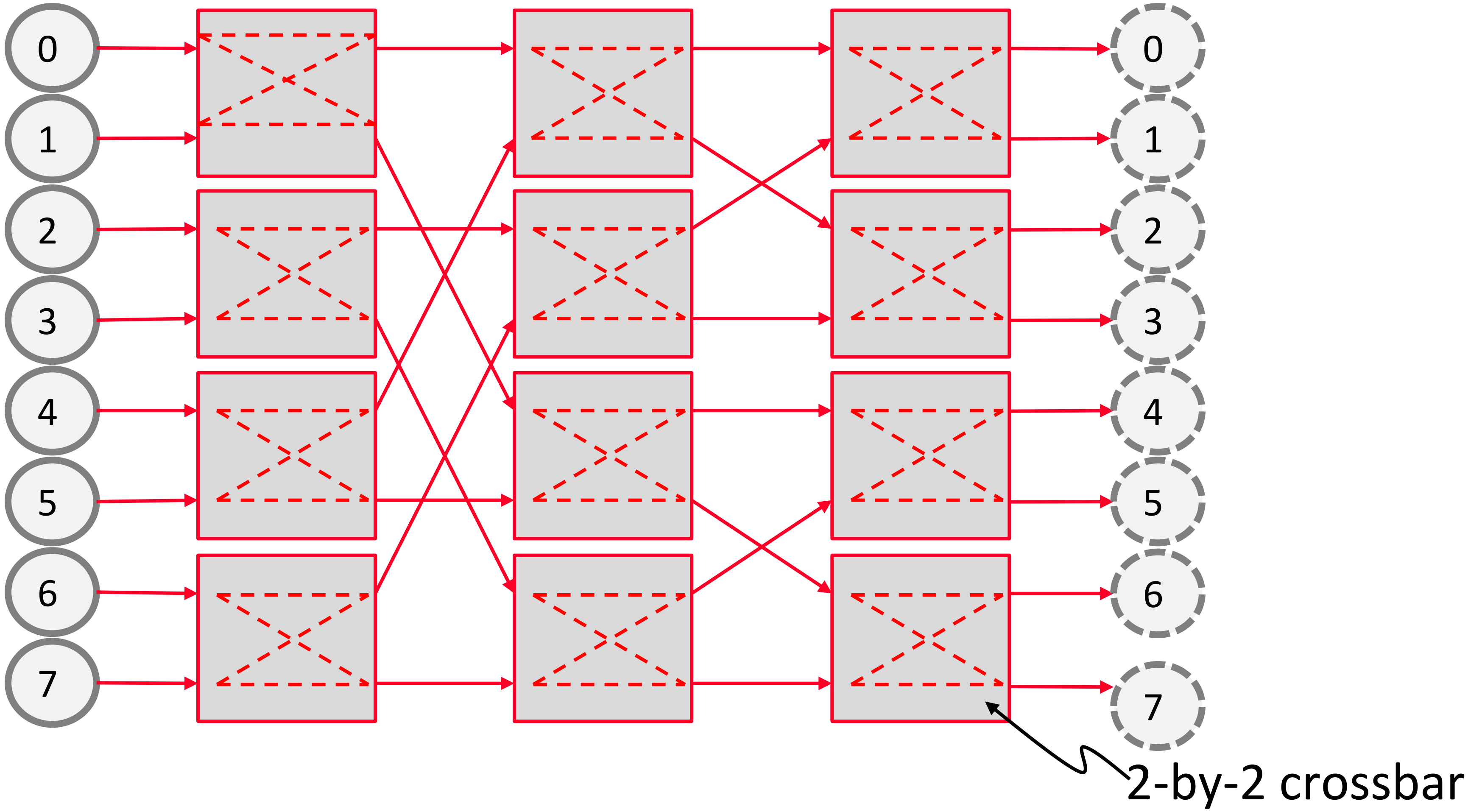


Multistage Networks



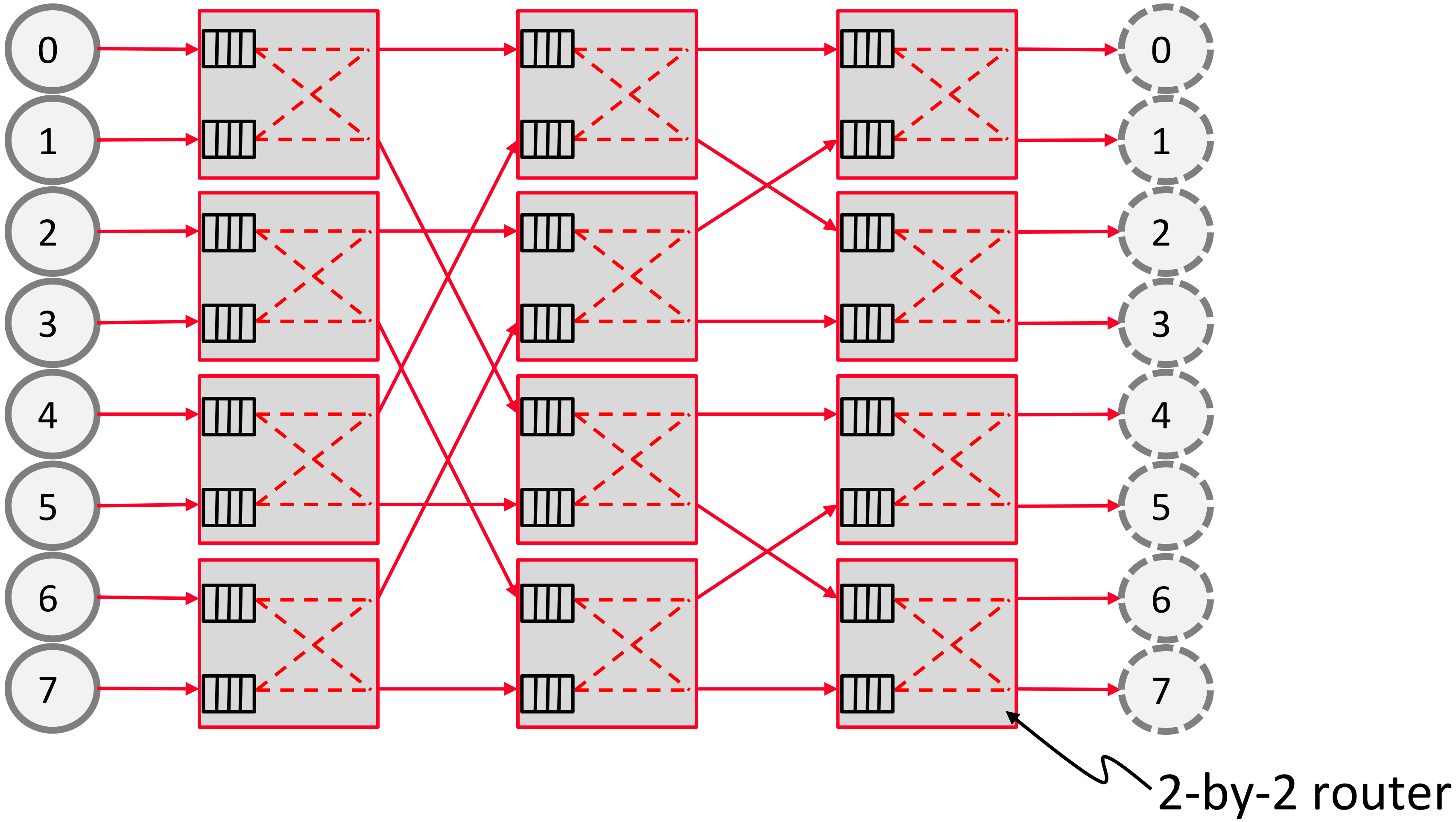
- A multistage network restricts concurrent Tx-Rx pairs vs. a crossbar
- But, it is less costly than crossbar, e.g., O(N logN) for Butterfly

Multistage Networks (Circuit Switched)



- Source-destination path completely set up (i.e., switches configured) to transmit data before data is transmitted – pre-configure switches
- No need for buffering since switching is static once path is set up

Multistage Networks (Packet Switched)



- Packets "hop" from router to router, pending availability of the next-required switch and buffer

Circuit vs. Packet Switching

- Circuit switching sets up full path before transmission
  - Establish route then send data
  - Noone else can use those links while "circuit" is set

  + faster arbitration

  + no buffering

  -- setting up and bringing down "path" takes time

  -- path cannot be used by multiple flows concurrently
- Packet switching performs routing per packet in each router
  - Route each packet individually (possibly via different paths)
  - If link is free, any packet can use it

  -- potentially slower --- must dynamically switch

  -- need handling contention (e.g., via buffering)

  + no setup, bring down time

  + more flexible, does not underutilize links

Switching vs. Topology

- Circuit/packet switching choice independent of topology
- It is a higher-level protocol on how a message gets sent to a destination

- However, some topologies are more amenable to circuit vs. packet switching

Heterogeneous Interconnects (in Tilera)

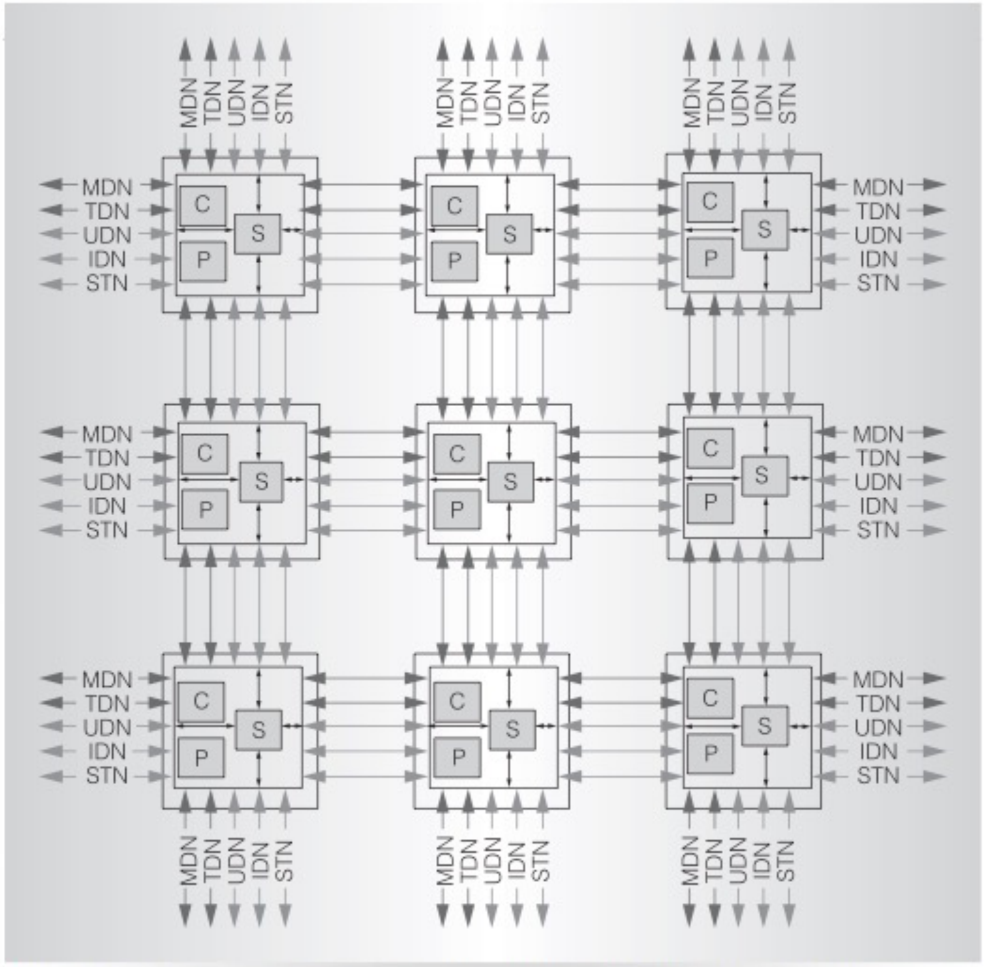


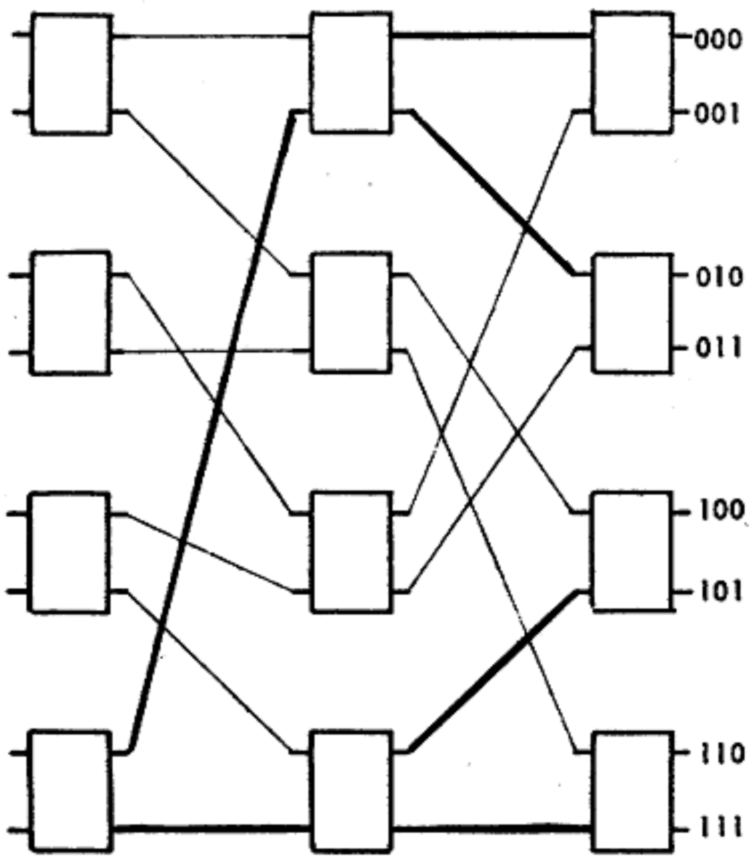
Figure 3. A 3 × 3 array of tiles connected by networks. (MDN: memory dynamic network; TDN: tile dynamic network; UDN: user dynamic network; IDN: I/O dynamic network; STN: static network.)

- Topology: 2D Mesh
- Five networks
- Four packet switched
  - Dimension order routing, wormhole flow control
  - TDN: Cache request packets
  - MDN: Response packets
  - IDN: I/O packets
  - UDN: Core to core messaging
- One circuit switched
  - STN: Low-latency, high-bandwidth static network
  - Streaming data

Wentzlaff et al., "On-Chip Interconnection Architecture of the Tile Processor," IEEE Micro 2007.

Another Multistage Network: Delta Network

- Single path from source to destination
- Each stage has different routers
- Proposed to replace costly crossbars as processor-memory interconnect
- Janak H. Patel, "Processor-Memory Interconnections for Multiprocessors," ISCA 1979.



8x8 Delta network

Another Multistage Network: Omega Network

- Single path from source to destination
- All stages are the same
- Used in NYU Ultracomputer
- Gottlieb et al. "The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer," IEEE Trans. On Comp., 1983.

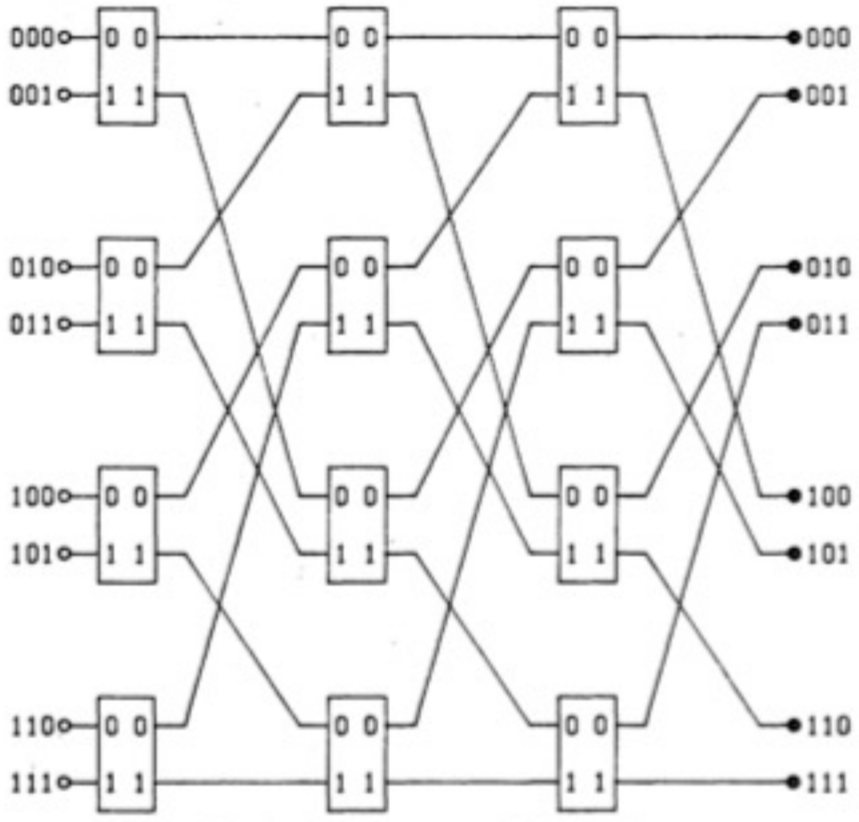


Fig. 2. Omega-network ($N = 8$).

Combining Operations in the Network

- Idea: Combine multiple operations on a shared memory location
- Example: Omega network switches combine fetch-and-add operations in NYU Ultracomputer
- Fetch-and-add(M, I): return M, replace M with M+I
  - Common when parallel processors modify a shared variable, e.g., obtain a chunk of the array
- Combining reduces synchronization latency

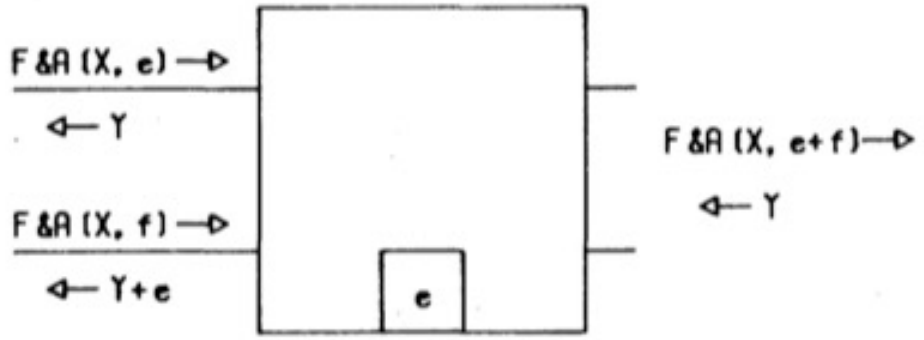


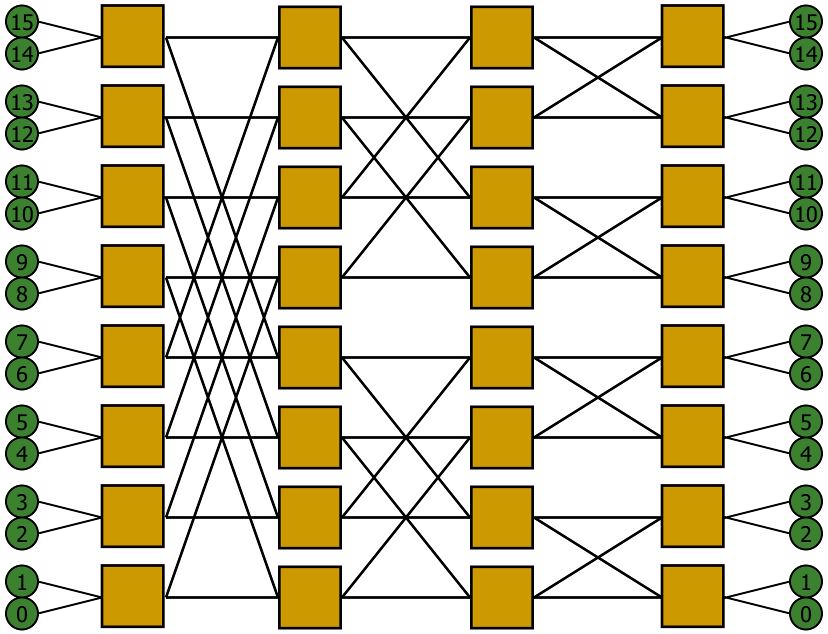
Fig. 3. Combining Fetch-and-Adds.

```
TestAndSet(V)
  {Temp ← V
      V ← TRUE}
  RETURN Temp.
```

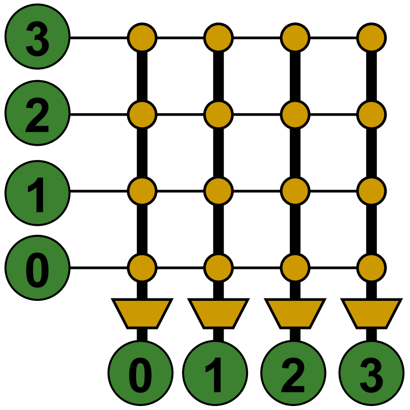
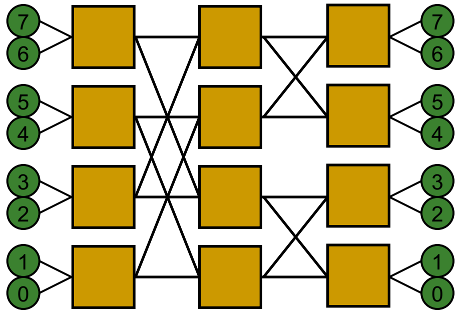
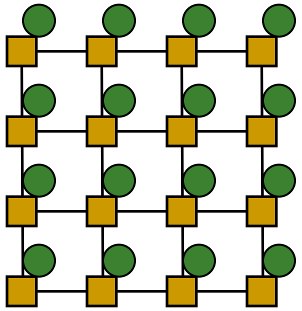
Fetch&OR(V, TRUE).

Butterfly

- Equivalent to Omega Network
- Indirect
- Used in BBN Butterfly
- Conflicts can cause "*tree saturation*"
  - Randomization of route selection helps



Review: Topologies

			
Topology	**Crossbar**	**Multistage Logarith.**	**Mesh**
Direct/Indirect	**Indirect**	**Indirect**	**Direct**
Blocking/ Non-blocking	**Non-blocking**	**Blocking**	**Blocking**
Cost	**$O(N^2)$**	**O(NlogN)**	**O(N)**
Latency	**O(1)**	**O(logN)**	**O(sqrt(N))**

Ring

Each node connected to exactly two other nodes. Nodes form a continuous pathway such that packets can reach any node.

+ Cheap: O(N) cost

- High latency: O(N)

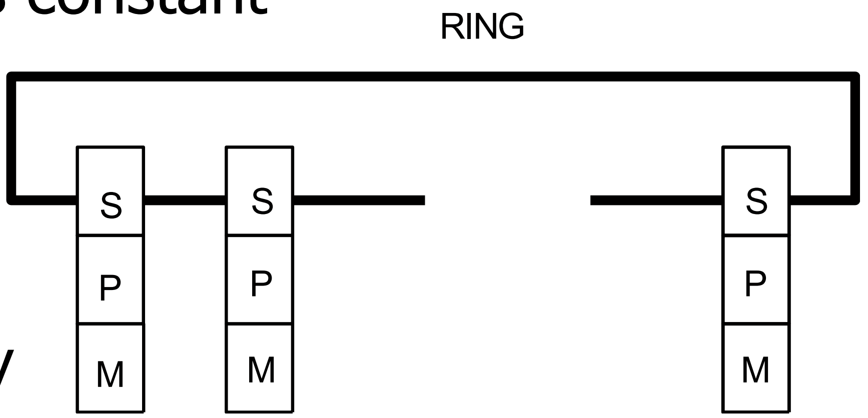
- Not easy to scale

    - Bisection bandwidth remains constant

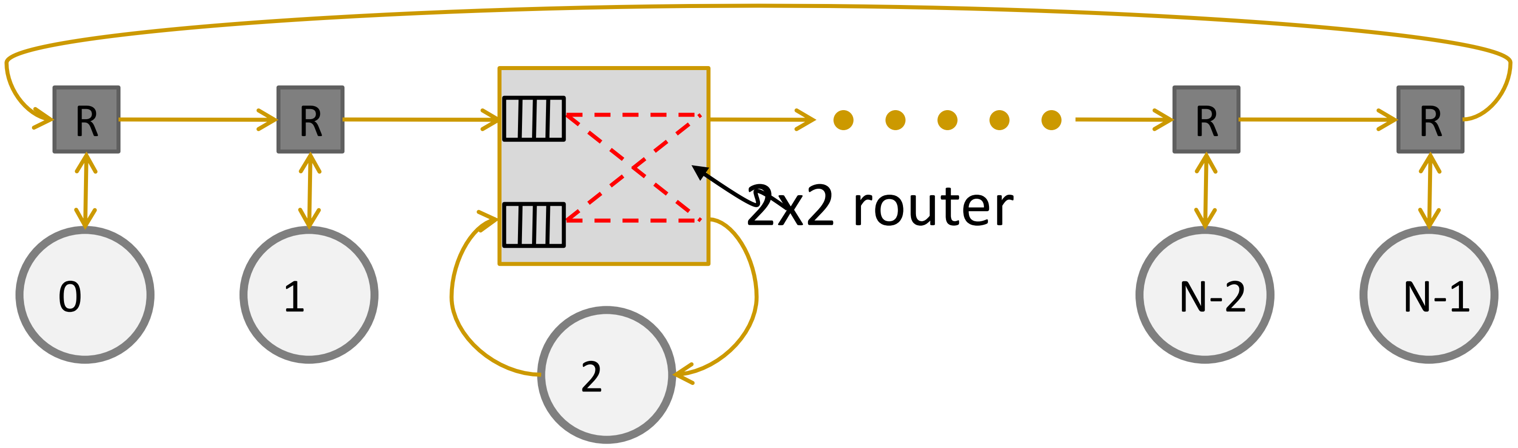
Used in Intel Haswell,

Intel Larrabee, IBM Cell,

many commercial systems today



Unidirectional Ring



- Single directional pathway
- Simple topology and implementation
  - Reasonable performance if N and performance needs (bandwidth & latency) still moderately low
  - O(N) cost
  - N/2 average hops; latency depends on utilization

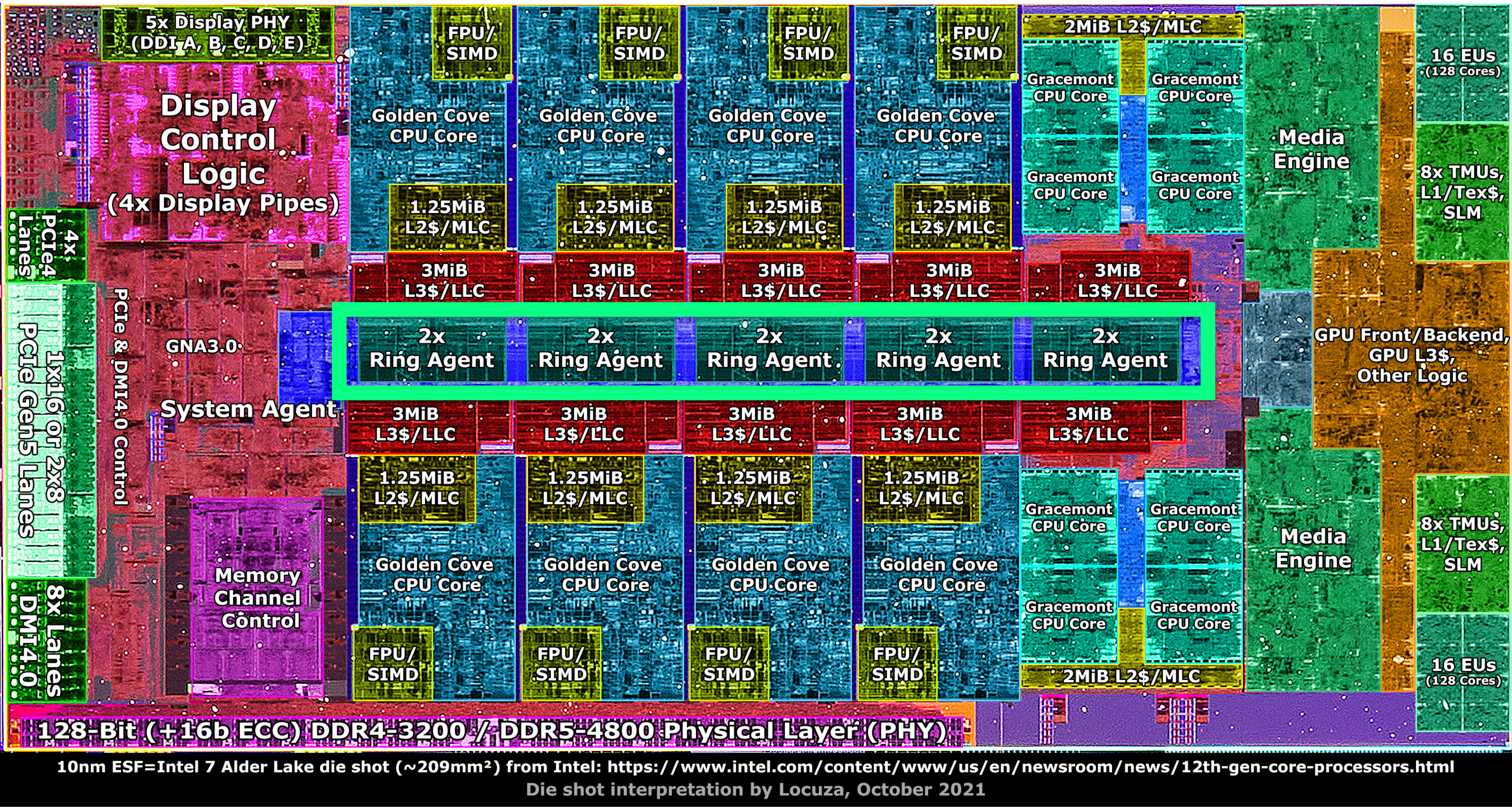
Bidirectional Rings

Multi-directional pathways, or multiple rings

+ Reduces latency
+ Improves scalability

- Slightly more complex injection policy (need to select which ring to inject a packet into)

Rings in Existing Systems

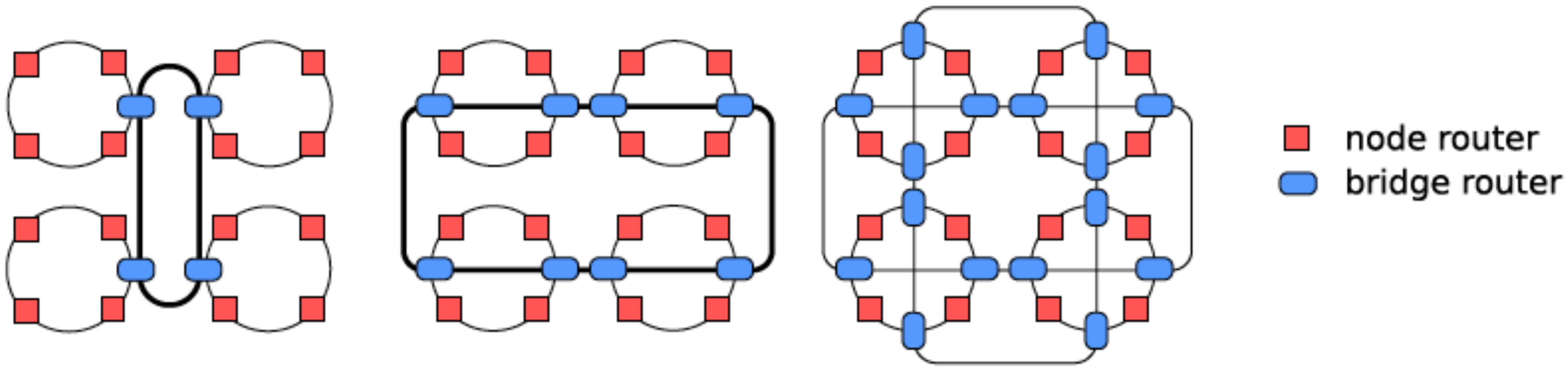


10nm ESF=Intel 7 Alder Lake die shot (~209mm²) from Intel: https://www.intel.com/content/www/us/en/newsroom/news/12th-gen-core-processors.html
Die shot interpretation by Locuza, October 2021

Intel Alder Lake, 2021

Source: https://twitter.com/Locuza_/status/1454152714930331652

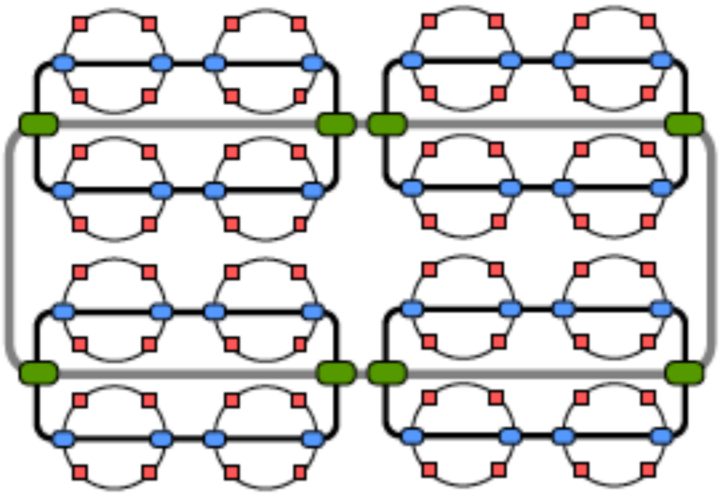
Hierarchical Rings



(a) 4-, 8-, and 16-bridge hierarchical ring topologies.

+ More scalable
+ Lower latency

- More complex



(b) Three-level hierarchy (8x8).

More on Hierarchical Rings

- Rachata Ausavarungnirun, Chris Fallin, Xiangyao Yu, Kevin Chang, Greg Nazario, Reetuparna Das, Gabriel Loh, and Onur Mutlu,
"Design and Evaluation of Hierarchical Rings with Deflection Routing"
*Proceedings of the 26th International Symposium on Computer Architecture and High Performance Computing (**SBAC-PAD**)*, Paris, France, October 2014. [Slides (pptx) (pdf)] [Source Code]

- Describes the design and implementation of a mostly-bufferless hierarchical ring

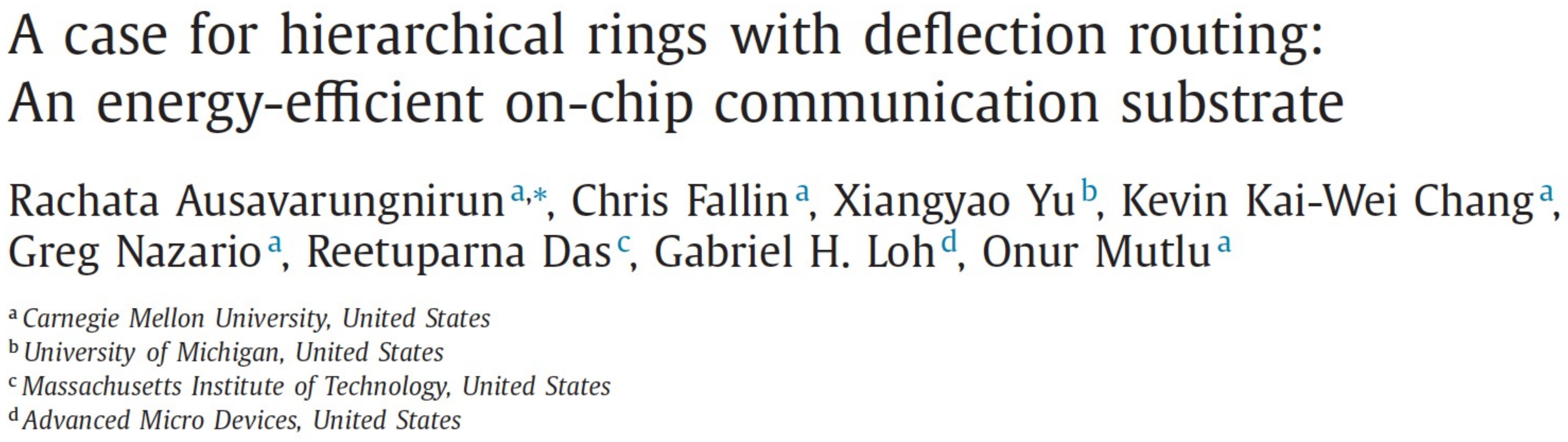
Design and Evaluation of Hierarchical Rings with Deflection Routing

Rachata Ausavarungnirun Chris Fallin Xiangyao Yu† Kevin Kai-Wei Chang
Greg Nazario Reetuparna Das§ Gabriel H. Loh‡ Onur Mutlu

Carnegie Mellon University §University of Michigan †MIT ‡Advanced Micro Devices, Inc.

More on Hierarchical Rings (II)

- Rachata Ausavarungnirun, Chris Fallin, Xiangyao Yu, Kevin Chang, Greg Nazario, Reetuparna Das, Gabriel Loh, and Onur Mutlu,
"A Case for Hierarchical Rings with Deflection Routing: An Energy-Efficient On-Chip Communication Substrate"
Parallel Computing *(**PARCO**)*, 2016. arXiv.org version, February 2016.

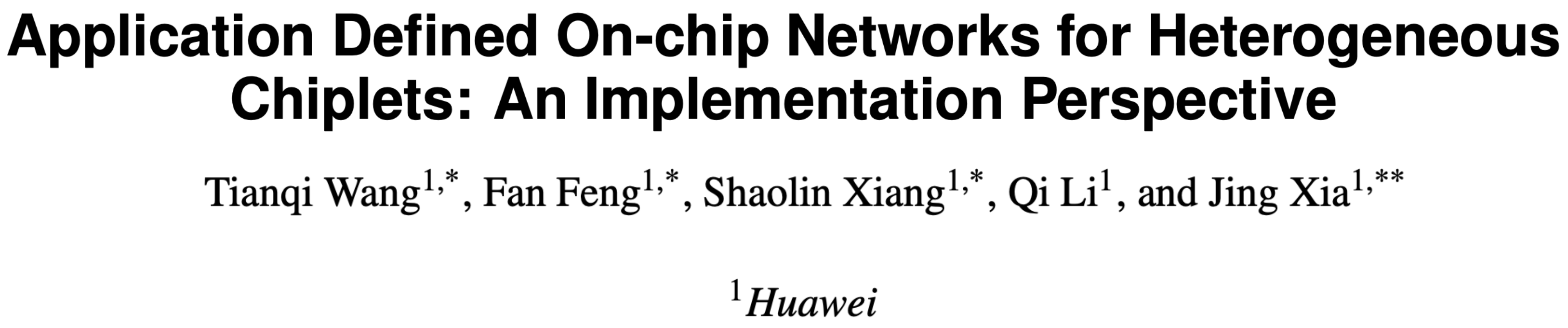


A case for hierarchical rings with deflection routing: An energy-efficient on-chip communication substrate

Rachata Ausavarungnirun[a,*], Chris Fallin[a], Xiangyao Yu[b], Kevin Kai-Wei Chang[a], Greg Nazario[a], Reetuparna Das[c], Gabriel H. Loh[d], Onur Mutlu[a]

[a] *Carnegie Mellon University, United States*
[b] *University of Michigan, United States*
[c] *Massachusetts Institute of Technology, United States*
[d] *Advanced Micro Devices, United States*

Hierarchical Rings in Real Systems

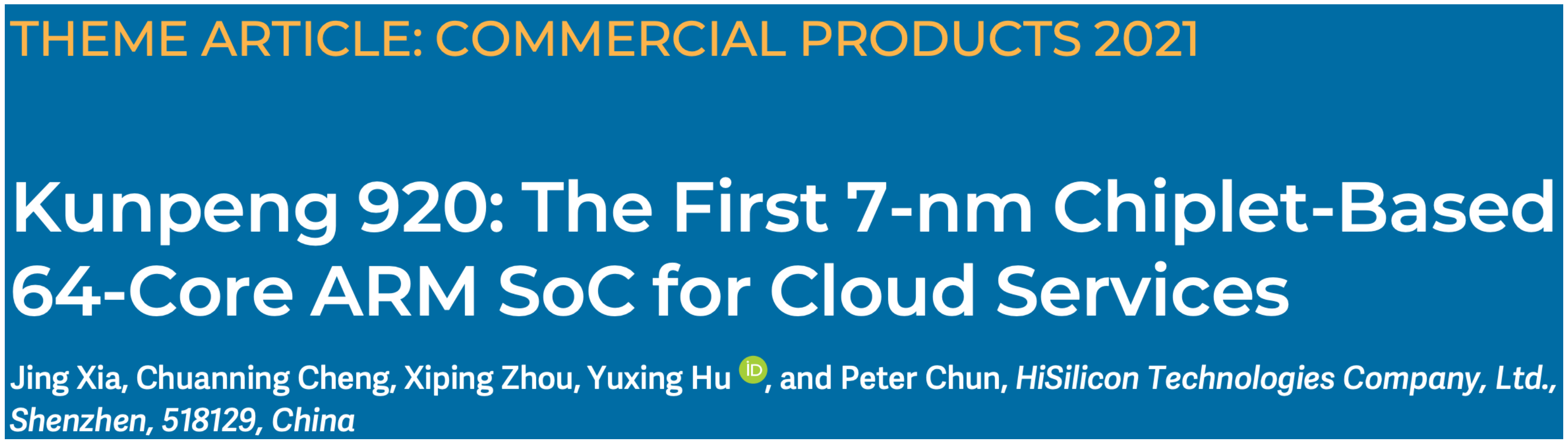


Application Defined On-chip Networks for Heterogeneous Chiplets: An Implementation Perspective

Tianqi Wang[1,*], Fan Feng[1,*], Shaolin Xiang[1,*], Qi Li[1], and Jing Xia[1,**]

[1]*Huawei*

https://ieeexplore.ieee.org/document/9773184/



THEME ARTICLE: COMMERCIAL PRODUCTS 2021

Kunpeng 920: The First 7-nm Chiplet-Based 64-Core ARM SoC for Cloud Services

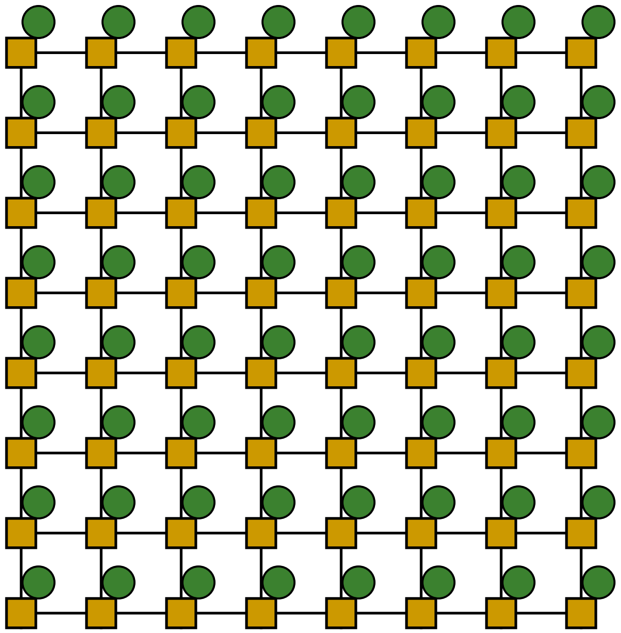
Jing Xia, Chuanning Cheng, Xiping Zhou, Yuxing Hu, and Peter Chun, *HiSilicon Technologies Company, Ltd., Shenzhen, 518129, China*

https://ieeexplore.ieee.org/abstract/document/9444893

Mesh

- Each node connected to 4 neighbors (N, E, S, W)
- O(N) cost
- Average latency: O(sqrt(N))
- Easy to layout on-chip: regular and equal-length links
- Path diversity: many ways to get from one node to another

- Used in Tilera 100-core
- And many on-chip network prototypes



Torus

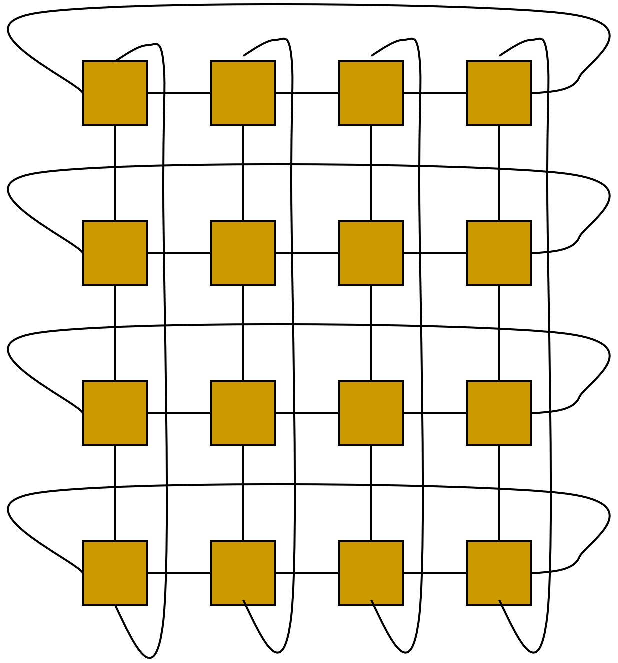
- Mesh is not symmetric on edges: performance very sensitive to placement of task on edge vs. middle
- Torus avoids this problem

+ Higher path diversity (and bisection bandwidth) than mesh

- Higher cost

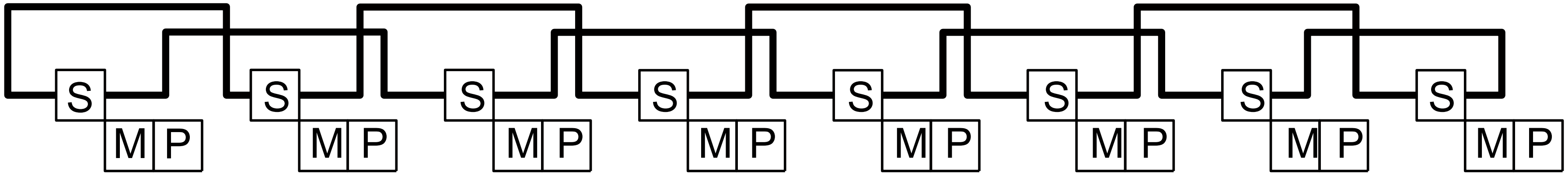
- Harder to lay out on-chip

  - Unequal link lengths



Torus, continued

- Weave nodes to make inter-node latencies ~constant



Trees

Planar, hierarchical topology

Latency: O(logN)

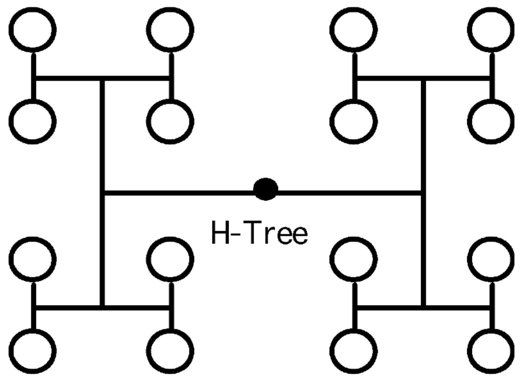
Good for local traffic

+ Cheap: O(N) cost

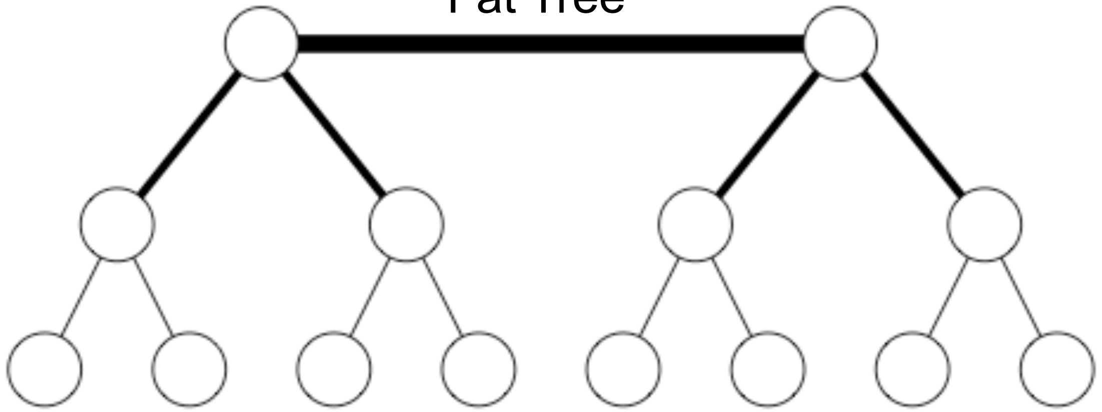
+ Easy to Layout

- Root can become a bottleneck

  Fat trees avoid this problem (CM-5)

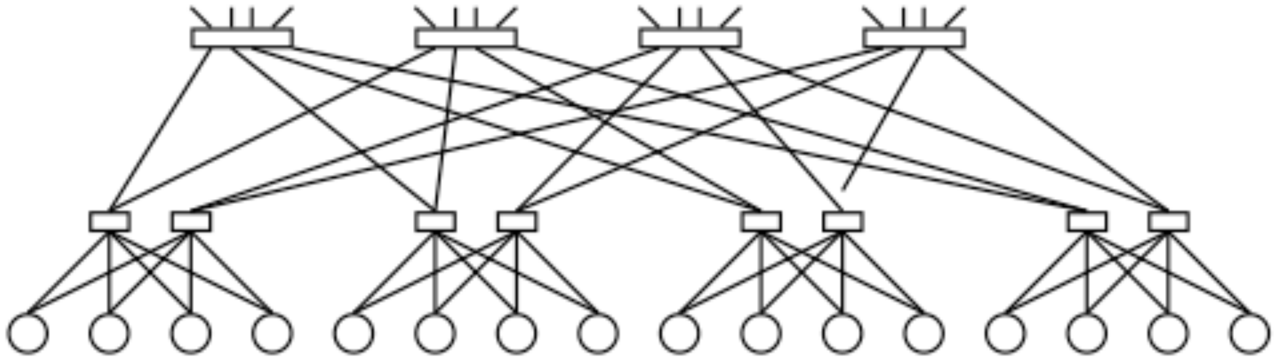


Fat Tree



CM-5 Fat Tree

- Fat tree based on 4x2 switches, packet switched
- Randomized routing on the way up
- Combining, multicast, reduction operators supported in hardware
  - Thinking Machines Corp., "The Connection Machine CM-5 Technical Summary," Jan. 1992.



CM-5 Thinned Fat Tree

CM-5 Fat Tree

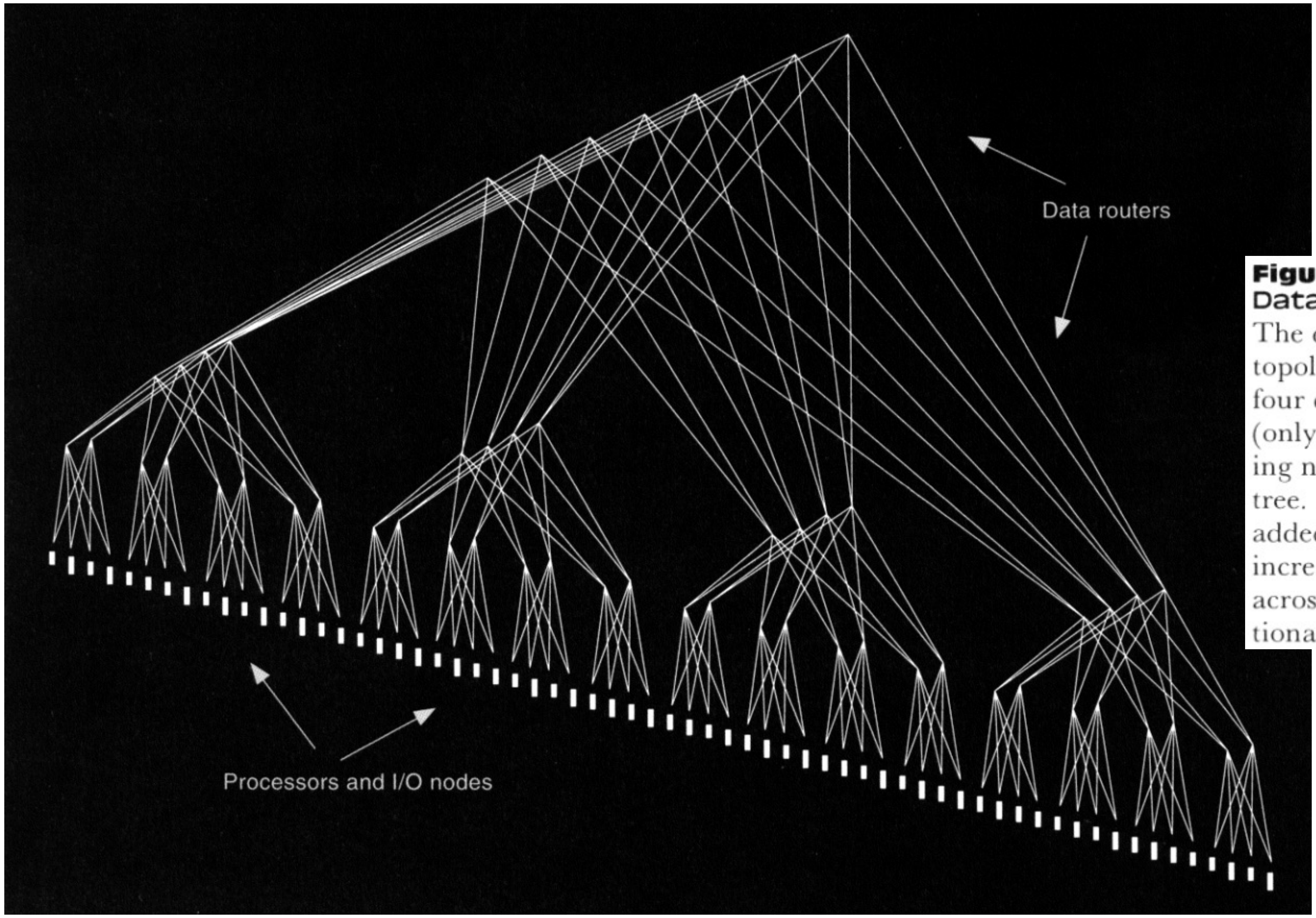
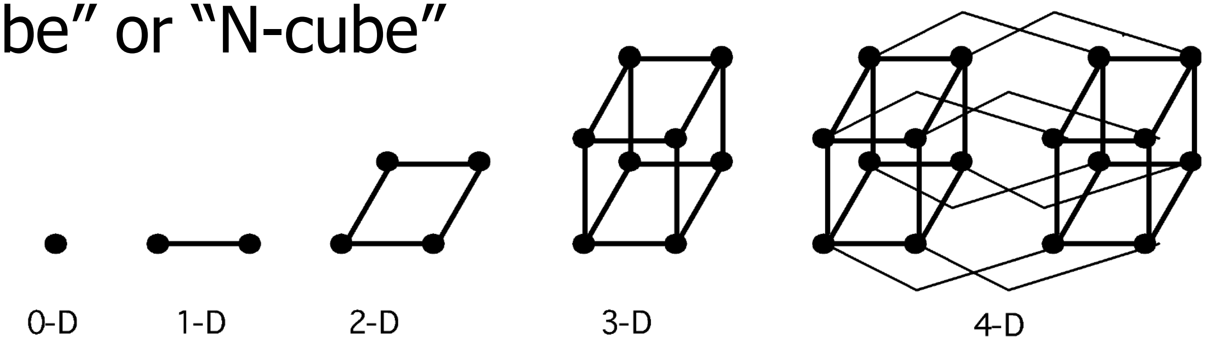


Figure 4.
Data network topology
The data network uses a fat tree topology. Each interior node connects four children to two or more parents (only two parents are shown). Processing nodes form the leaf nodes of the tree. As more processing nodes are added, the number of levels of the tree increases, allowing the total bandwidth across the network to increase proportionally to the number of processors.

Hillis & Tucker, "The CM-5 Connection Machine: A Scalable Supercomputer," CACM 1993.

Hypercube

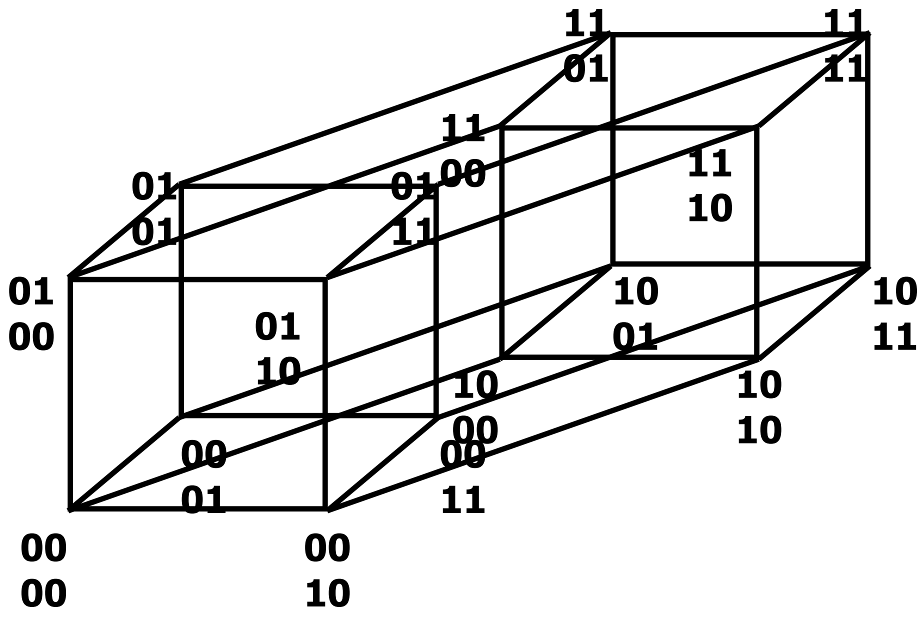
- "N-dimensional cube" or "N-cube"



- Latency: O(logN)
- Radix: O(logN)
- #links: O(NlogN)

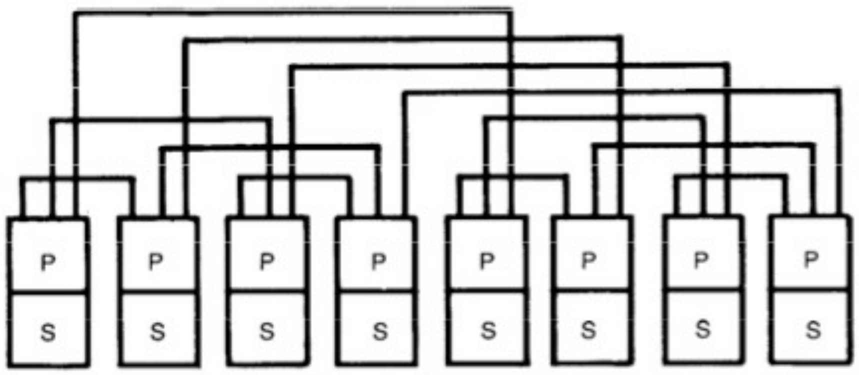
+ Low latency

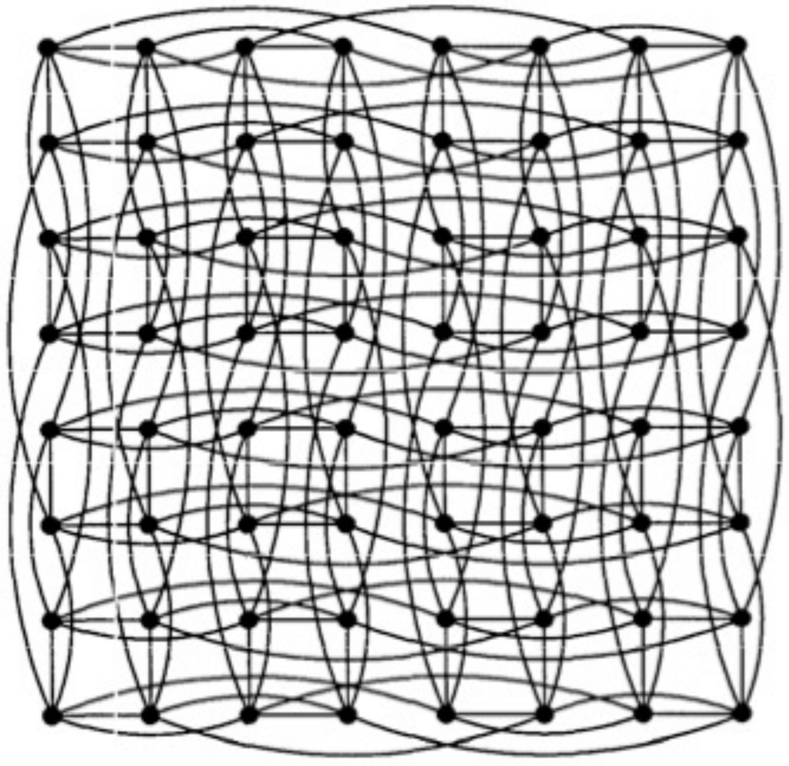
- Hard to lay out in 2D/3D



Caltech Cosmic Cube

- 64-node message passing machine
- Seitz, "The Cosmic Cube," CACM 1985.

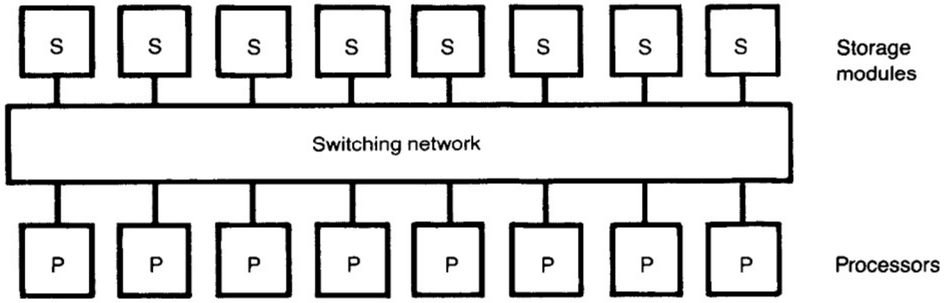




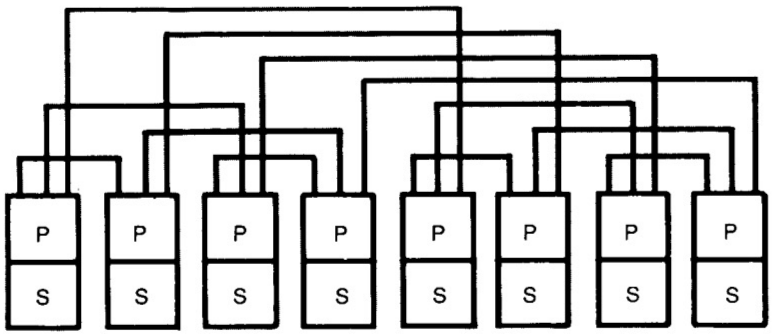
A hypercube connects $N = 2^n$ small computers, called nodes, through point-to-point communication channels in the Cosmic Cube. Shown here is a two-dimensional projection of a six-dimensional hypercube, or binary 6-cube, which corresponds to a 64-node machine.

FIGURE 1. A Hypercube (also known as a binary cube or a Boolean *n*-cube)

Caltech Cosmic Cube Motivation



(a) Most multiprocessors are structured with a switching network, either a crossbar connection of buses or a multistage routing network, between the processors and storage. The switching network introduces a latency in the communication between processors and storage, and does not scale well to large sizes. Communication between processes running concurrently in different processors occurs through shared variables and common access to one large address space.



(b) Message-passing multicomputer systems retain a physically close and fast connection between processors and their associated storage. The concurrent computers (nodes) can send messages through a network of communication channels. The network shown here is a three-dimensional cube, which is a small version of the communication plan used in six dimensions in the 64-node Cosmic Cube.

NOTE: Actual machines need not follow one model or the other absolutely: Various hybrids are possible.

FIGURE 2. A Comparison of Shared-Storage Multiprocessors and Message-Passing Machines

Seitz, "The Cosmic Cube," CACM 1985.

Routing

Routing Mechanism

- Arithmetic
  - Simple arithmetic to determine route in regular topologies
  - Dimension order routing in meshes/tori

- Source Based
  - Source specifies output port for each switch in route

  + Simple switches
    - no control state: strip output port off header

  - Large header

- Table Lookup Based
  - Index into table for output port

  + Small header

  - More complex switches

Routing Algorithm

- Three Types
  - Deterministic: always chooses the same path for a communicating source-destination pair
  - Oblivious: chooses different paths, without considering network state
  - Adaptive: can choose different paths, adapting to the state of the network

- How to adapt
  - Local/global feedback
  - Minimal or non-minimal paths

Deterministic Routing

- All packets between the same (source, dest) pair take the same path

- Dimension-order routing
  - First traverse dimension X, then traverse dimension Y
  - E.g., XY routing (used in Cray T3D, and many on-chip networks)

+ Simple

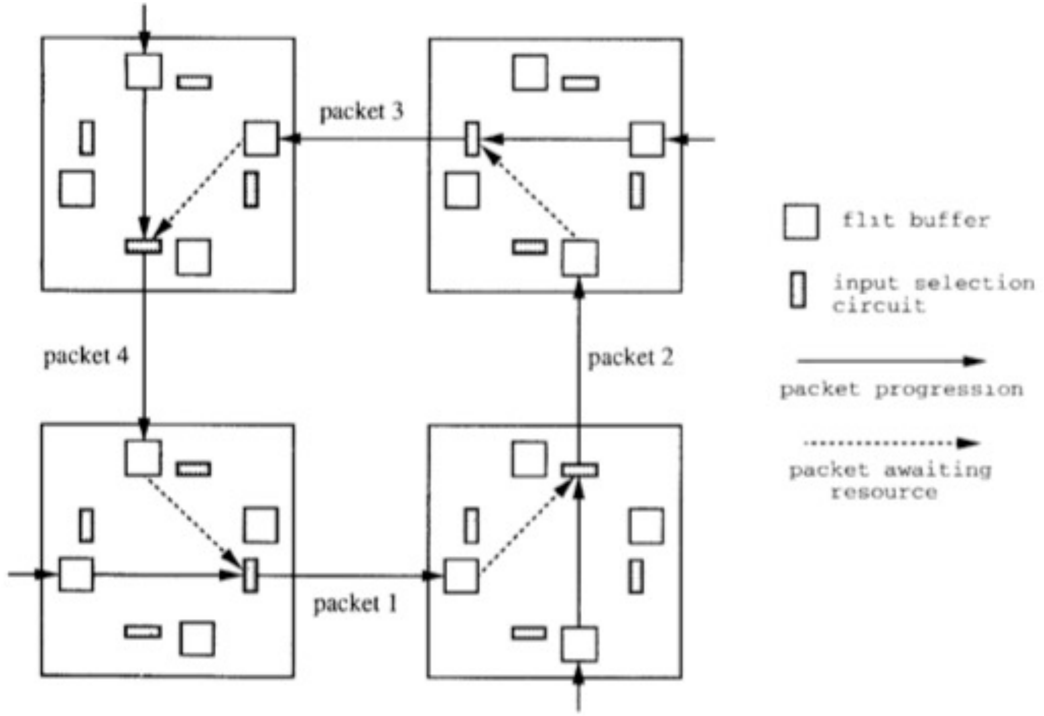
+ Deadlock freedom (no cycles in resource allocation)

- Could lead to high contention

- Does not exploit path diversity

Deadlock

- No forward progress
- Caused by circular dependencies on resources
- Each packet waits for a buffer occupied by another packet downstream



Handling Deadlock

- Avoid cycles in routing
  - Dimension order routing
    - Cannot build a circular dependency
  - Restrict the “turns” each packet can take
- Avoid deadlock by adding more buffering (escape paths)
- Detect and break deadlock
  - Preemption of buffers

Turn Model to Avoid Deadlock

- Idea
  - Analyze directions in which packets can turn in the network
  - Determine the cycles that such turns can form
  - Prohibit just enough turns to break possible cycles
- Glass and Ni, "The Turn Model for Adaptive Routing," ISCA 1992.

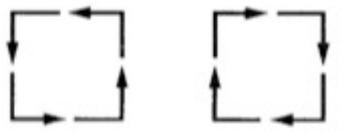


FIG. 2. The possible turns and simple cycles in a two-dimensional mesh.

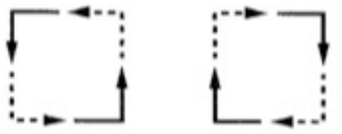


FIG. 3. The four turns allowed by the *xy* routing algorithm.

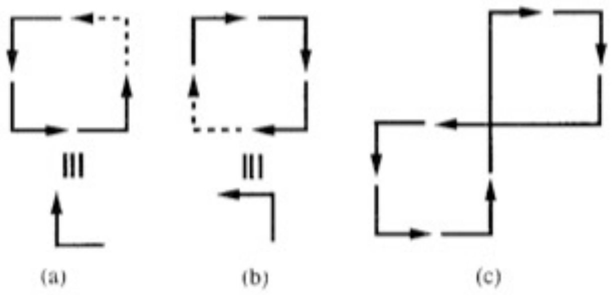


FIG. 4. Six turns that complete the cycles and allow deadlock.

Oblivious Routing: Valiant's Algorithm

- Goal: Balance network load
- Idea: Randomly choose an intermediate destination, route to it first, then route from there to destination
    - Between source-intermediate and intermediate-dest, can use dimension order routing

+ Randomizes/balances network load

- Non minimal (packet latency can increase)

- Optimizations:
    - Do this on high load
    - Restrict the intermediate node to be close (in the same quadrant)

Valiant, "A Scheme for Fast Parallel Communication," SIAM Journal of Computing, 1982.

More on Valiant's Algorithm

- Valiant and Brebner, "Universal Schemes for Parallel Communication," STOC 1981.
- Valiant, "A Scheme for Fast Parallel Communication," SIAM Journal of Computing, 1982.

Adaptive Routing

- Minimal adaptive
  - Router uses network state (e.g., downstream buffer occupancy) to pick which "productive" output port to send a packet to
  - Productive output port: port that gets the packet closer to its destination

  + Aware of local congestion

  - Minimality restricts achievable link utilization (load balance)

- Non-minimal (fully) adaptive
  - "Misroute" packets to non-productive output ports based on network state

  + Can achieve better network utilization and load balance

  - Need to guarantee livelock freedom

More on Adaptive Routing

- Can avoid faulty links/routers

- Idea: Route around faults

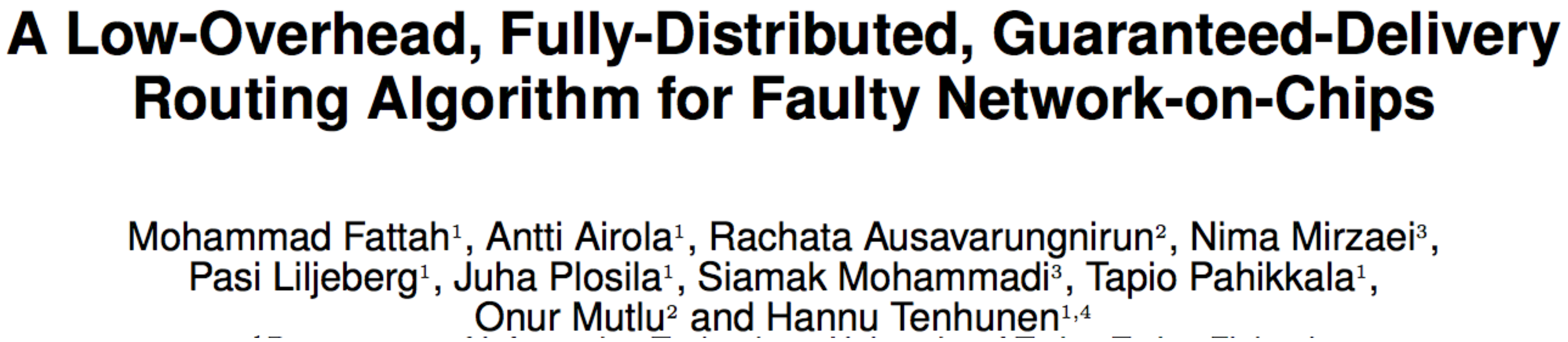
+ Deterministic routing cannot handle faulty components
- Need to change the routing table to disable faulty routes
  - Assuming the faulty link/router is detected

One relatively recent example:

Fattah et al., **"A Low-Overhead, Fully-Distributed, Guaranteed-Delivery Routing Algorithm for Faulty Network-on-Chips"**, NOCS 2015.

Fault Tolerance & Guaranteed Delivery

- Mohammad Fattah, Antti Airola, Rachata Ausavarungnirun, Nima Mirzaei, Pasi Liljeberg, Juha Plosila, Siamak Mohammadi, Tapio Pahikkala, Onur Mutlu, and Hannu Tenhunen, **"A Low-Overhead, Fully-Distributed, Guaranteed-Delivery Routing Algorithm for Faulty Network-on-Chips"**
  Proceedings of the 9th ACM/IEEE International Symposium on Networks on Chip (***NOCS***), Vancouver, BC, Canada, September 2015.
  [Slides (pptx) (pdf)]
  [Source Code]
  One of the three papers nominated for the Best Paper Award by the Program Committee.



A Low-Overhead, Fully-Distributed, Guaranteed-Delivery Routing Algorithm for Faulty Network-on-Chips

Mohammad Fattah[1], Antti Airola[1], Rachata Ausavarungnirun[2], Nima Mirzaei[3], Pasi Liljeberg[1], Juha Plosila[1], Siamak Mohammadi[3], Tapio Pahikkala[1], Onur Mutlu[2] and Hannu Tenhunen[1,4]

Buffering and Flow Control

Recall: Circuit vs. Packet Switching

- Circuit switching sets up full path before transmission
  - Establish route then send data
  - Noone else can use those links while "circuit" is set

  + faster arbitration

  + no buffering

  -- setting up and bringing down "path" takes time

  -- path cannot be used by multiple flows concurrently
- Packet switching performs routing per packet in each router
  - Route each packet individually (possibly via different paths)
  - If link is free, any packet can use it

  -- potentially slower --- must dynamically switch

  -- need handling contention (e.g., via buffering)

  + no setup, bring down time

  + more flexible, does not underutilize links

Recall: Heterogeneous Interconnects (in Tilera)

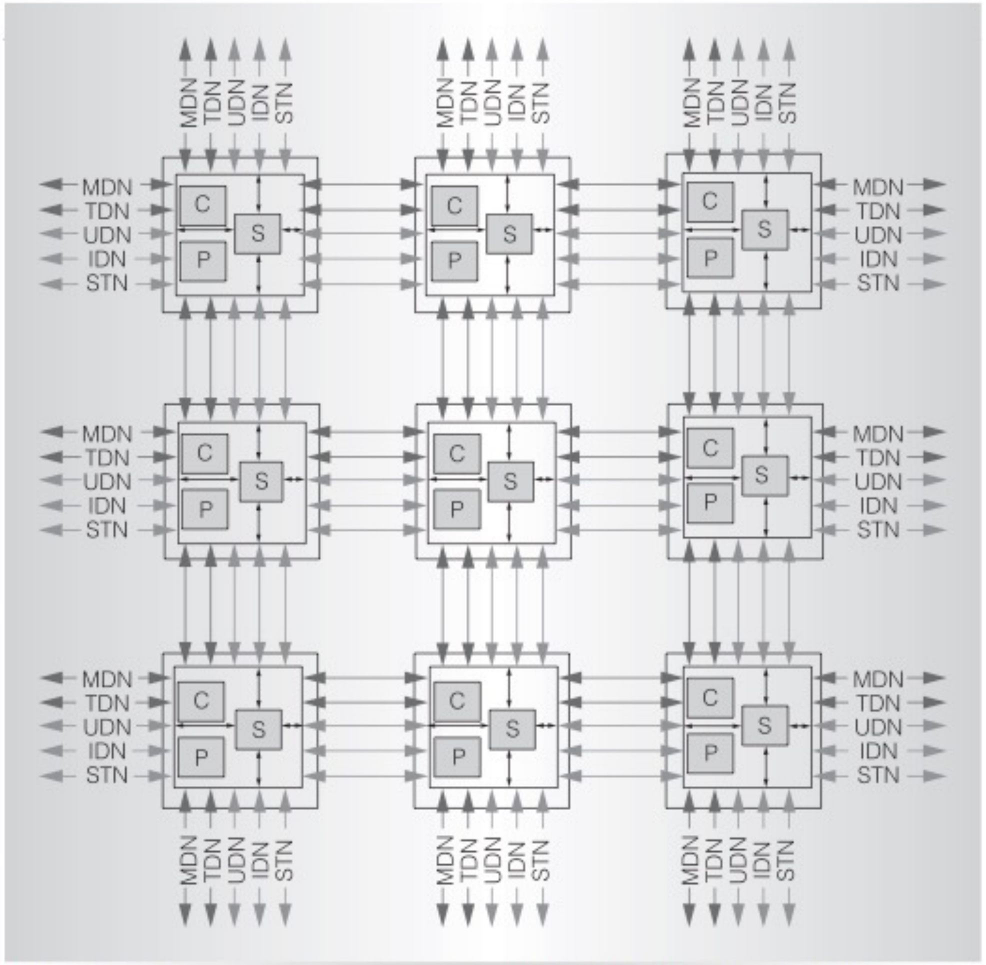


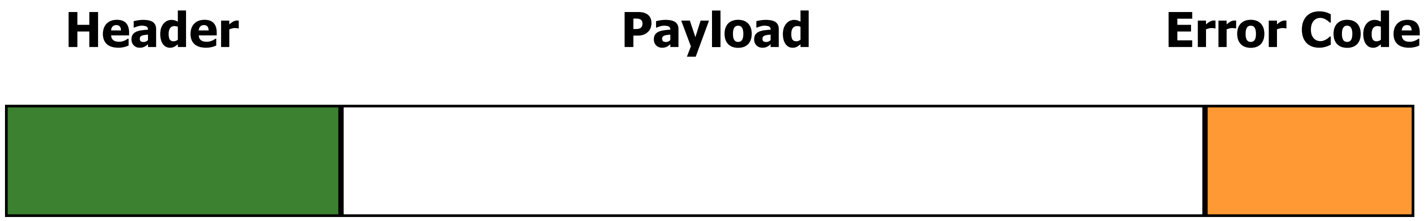
Figure 3. A 3 × 3 array of tiles connected by networks. (MDN: memory dynamic network; TDN: tile dynamic network; UDN: user dynamic network; IDN: I/O dynamic network; STN: static network.)

- Topology: 2D Mesh
- Five networks
- Four packet switched
  - Dimension order routing, wormhole flow control
  - TDN: Cache request packets
  - MDN: Response packets
  - IDN: I/O packets
  - UDN: Core to core messaging
- One circuit switched
  - STN: Low-latency, high-bandwidth static network
  - Streaming data

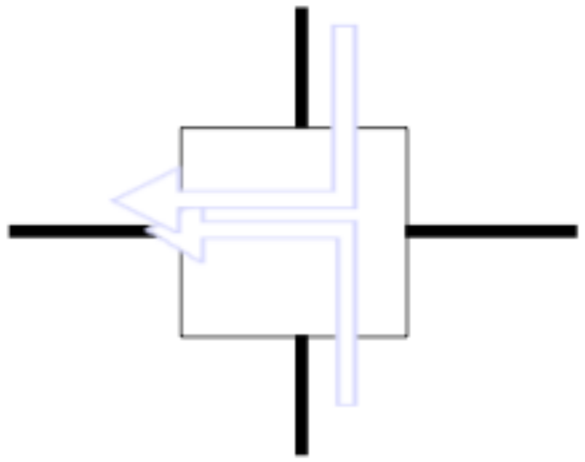
Wentzlaff et al., "On-Chip Interconnection Architecture of the Tile Processor," IEEE Micro 2007.

Packet Switched Networks: Packet Format

- Header
  - routing and control information
- Payload
  - carries data (non HW specific information)
  - can be further divided (framing, protocol stacks…)
- Error Code
  - generally at tail of packet so it can be generated on the way out



Handling Contention



- Two packets trying to use the same link at the same time
- What do you do?
  - Buffer one
  - Drop one
  - Misroute one (deflection)
- Tradeoffs?

Flow Control Methods

- Circuit switching
- Bufferless (Packet/flit based)
- Store and forward (Packet based)
- Virtual cut through (Packet based)
- Wormhole (Flit based)

Circuit Switching Revisited

- Resource allocation granularity is large

- Idea: Pre-allocate resources across multiple switches for a given "flow"
- Need to send a probe to set up the path for pre-allocation

+ No need for buffering

+ No contention (flow's performance is isolated)

+ Can handle arbitrary message sizes

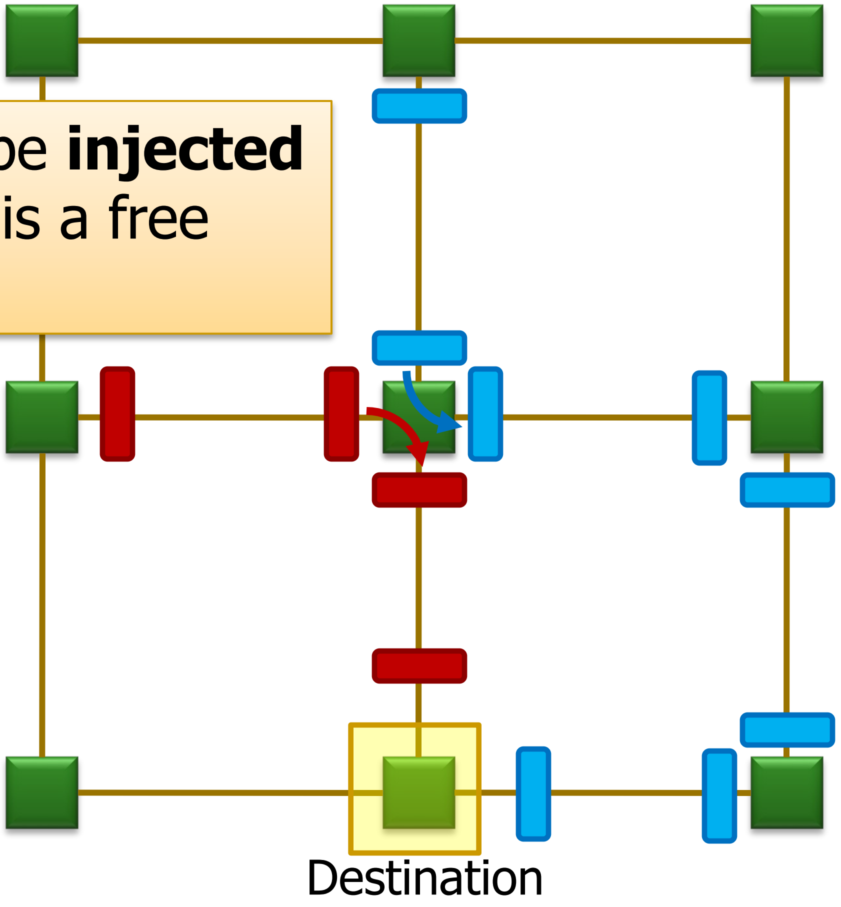
- Lower link utilization: two flows cannot use the same link

- Handshake overhead to set up a "circuit"

Bufferless Deflection Routing

- **Key idea**: Packets are never buffered in the network. When two packets contend for the same link, one is deflected.[1]

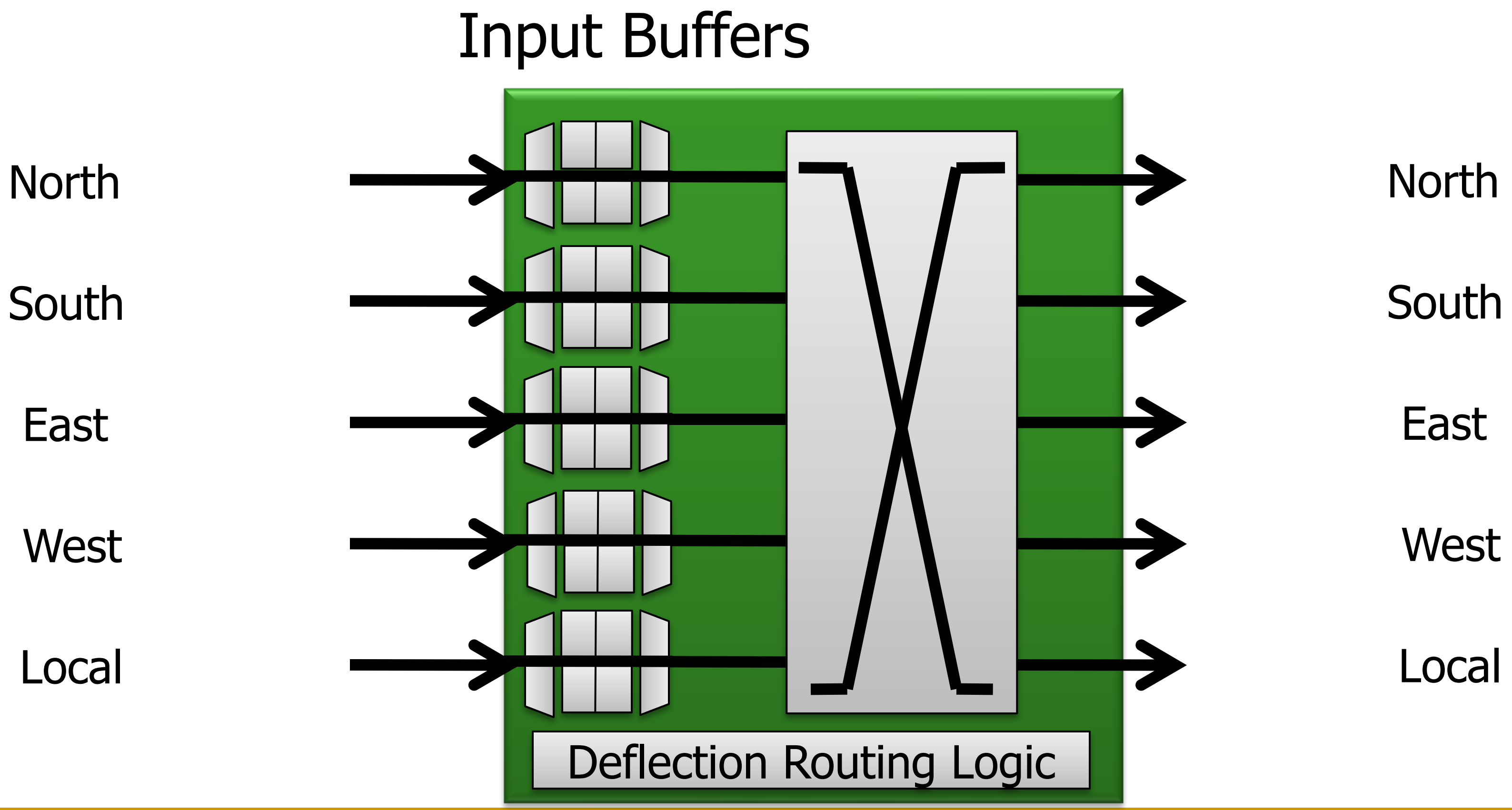
New traffic can be **injected** whenever there is a free output link.



[1]Baran, "On Distributed Communication Networks." RAND Tech. Report., 1962 / IEEE Trans.Comm., 1964.

Bufferless Deflection Routing

- Input buffers are eliminated: packets are buffered in **pipeline latches** and on **network links**



Moscibroda and Mutlu, "A Case for Bufferless Routing in On-Chip Networks," ISCA 2009. 83

Issues In Bufferless Deflection Routing

- Livelock

- Resulting Router Complexity

- Performance & Congestion at High Loads

- Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, and Onur Mutlu,
  "Bufferless and Minimally-Buffered Deflection Routing"
  Invited Book Chapter in Routing Algorithms in Networks-on-Chip, pp. 241-275, Springer, 2014.

Bufferless Deflection Routing in NoCs

- Thomas Moscibroda and Onur Mutlu,
  "A Case for Bufferless Routing in On-Chip Networks"
  Proceedings of the 36th International Symposium on Computer Architecture (***ISCA***), pages 196-207, Austin, TX, June 2009. Slides (pptx)

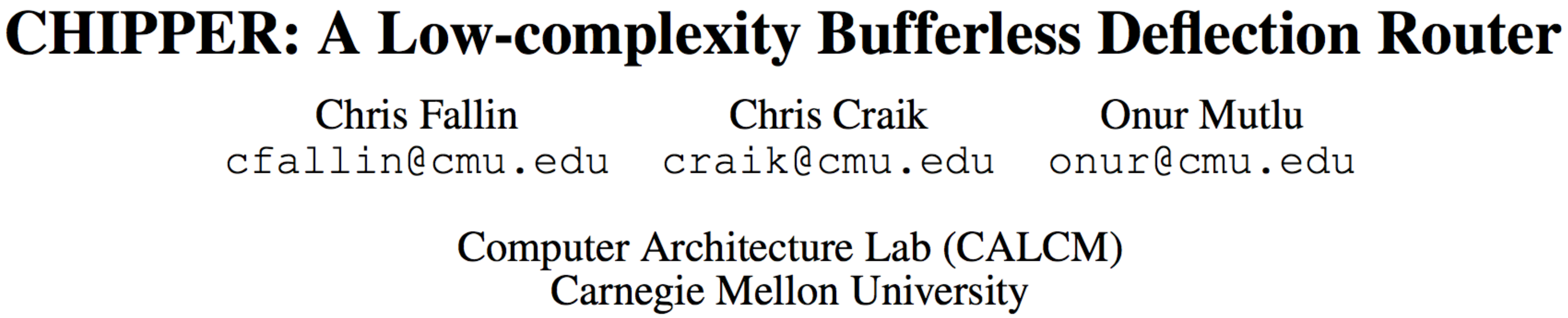
A Case for Bufferless Routing in On-Chip Networks

Thomas Moscibroda
Microsoft Research
moscitho@microsoft.com

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu

Low-Complexity Bufferless Routing

- Chris Fallin, Chris Craik, and Onur Mutlu,
  "CHIPPER: A Low-Complexity Bufferless Deflection Router"
  Proceedings of the 17th International Symposium on High-Performance Computer Architecture (***HPCA***), pages 144-155, San Antonio, TX, February 2011. Slides (pptx)



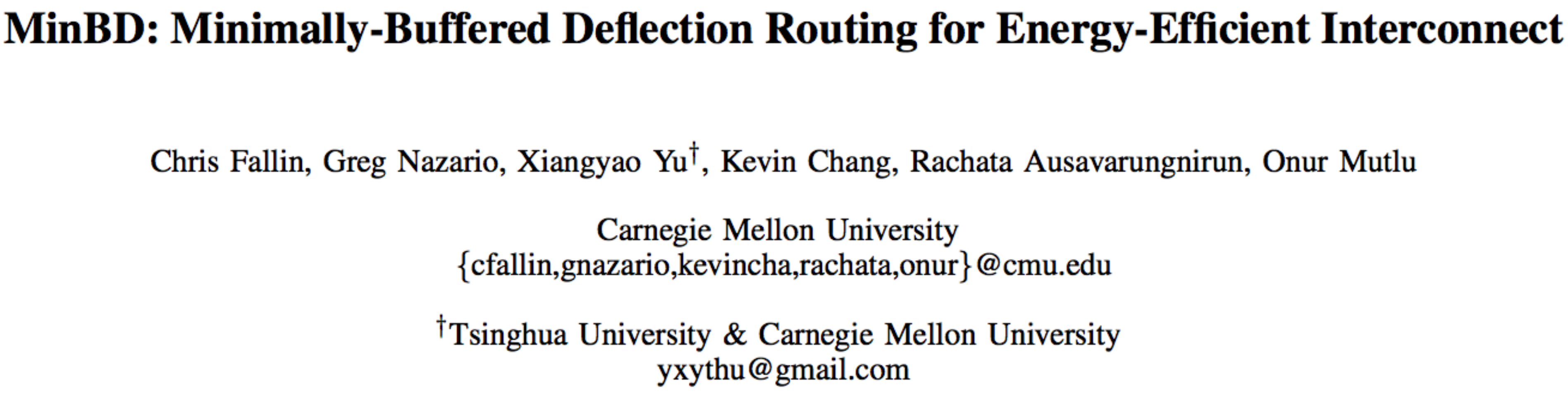
CHIPPER: A Low-complexity Bufferless Deflection Router

Chris Fallin cfallin@cmu.edu
Chris Craik craik@cmu.edu
Onur Mutlu onur@cmu.edu

Computer Architecture Lab (CALCM)
Carnegie Mellon University

Minimally-Buffered Deflection Routing

- Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, and Onur Mutlu,
"MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect"
*Proceedings of the 6th ACM/IEEE International Symposium on Networks on Chip (**NOCS**)*, Lyngby, Denmark, May 2012. Slides (pptx) (pdf)
One of the five papers nominated for the Best Paper Award by the Program Committee.



MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect

Chris Fallin, Greg Nazario, Xiangyao Yu[†], Kevin Chang, Rachata Ausavarungnirun, Onur Mutlu

Carnegie Mellon University
{cfallin,gnazario,kevincha,rachata,onur}@cmu.edu

[†]Tsinghua University & Carnegie Mellon University
yxythu@gmail.com

"Bufferless" Hierarchical Rings

- Rachata Ausavarungnirun, Chris Fallin, Xiangyao Yu, Kevin Chang, Greg Nazario, Reetuparna Das, Gabriel Loh, and Onur Mutlu,
"Design and Evaluation of Hierarchical Rings with Deflection Routing"
*Proceedings of the 26th International Symposium on Computer Architecture and High Performance Computing (**SBAC-PAD**)*, Paris, France, October 2014. [Slides (pptx) (pdf)] [Source Code]

- Describes the design and implementation of a mostly-bufferless hierarchical ring

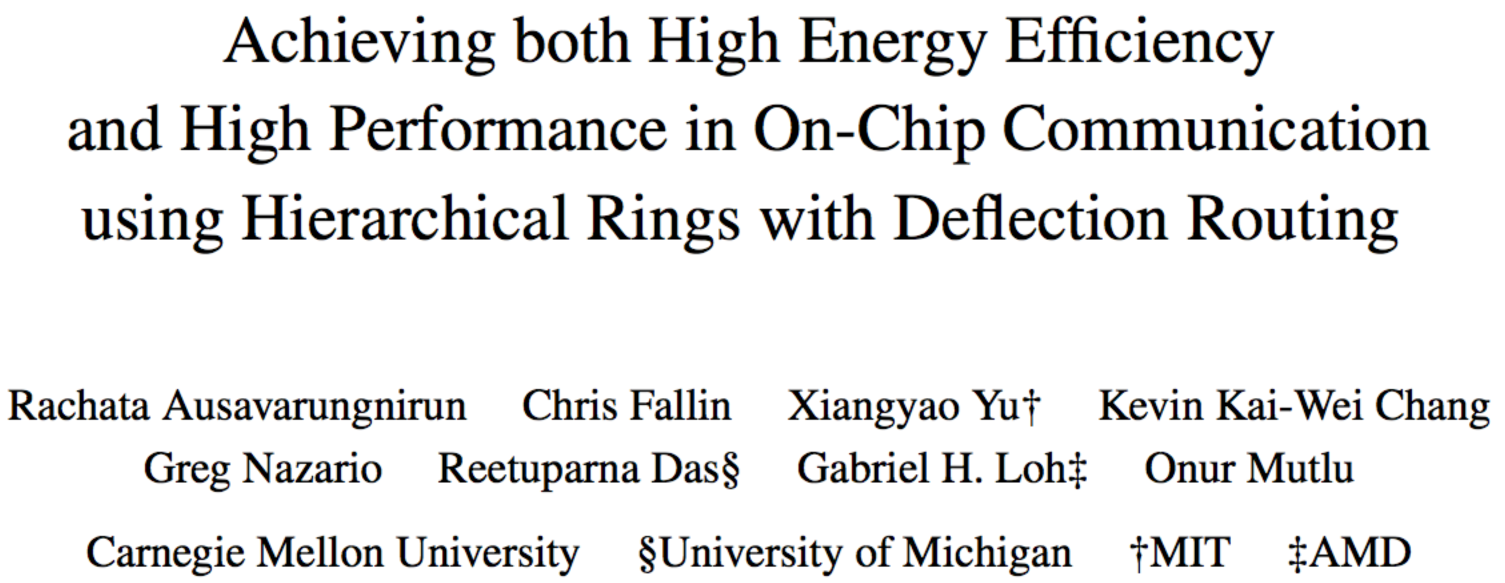
Design and Evaluation of Hierarchical Rings with Deflection Routing

Rachata Ausavarungnirun Chris Fallin Xiangyao Yu† Kevin Kai-Wei Chang
Greg Nazario Reetuparna Das§ Gabriel H. Loh‡ Onur Mutlu

Carnegie Mellon University §University of Michigan †MIT ‡Advanced Micro Devices, Inc.

“Bufferless” Hierarchical Rings (II)

- Rachata Ausavarungnirun, Chris Fallin, Xiangyao Yu, Kevin Chang, Greg Nazario, Reetuparna Das, Gabriel Loh, and Onur Mutlu, **"A Case for Hierarchical Rings with Deflection Routing: An Energy-Efficient On-Chip Communication Substrate"** ***Parallel Computing*** ***(PARCO)***, 2016.
  - arXiv.org version, February 2016.



Achieving both High Energy Efficiency and High Performance in On-Chip Communication using Hierarchical Rings with Deflection Routing

Rachata Ausavarungnirun Chris Fallin Xiangyao Yu† Kevin Kai-Wei Chang
Greg Nazario Reetuparna Das§ Gabriel H. Loh‡ Onur Mutlu

Carnegie Mellon University §University of Michigan †MIT ‡AMD

A Review of Bufferless Interconnects

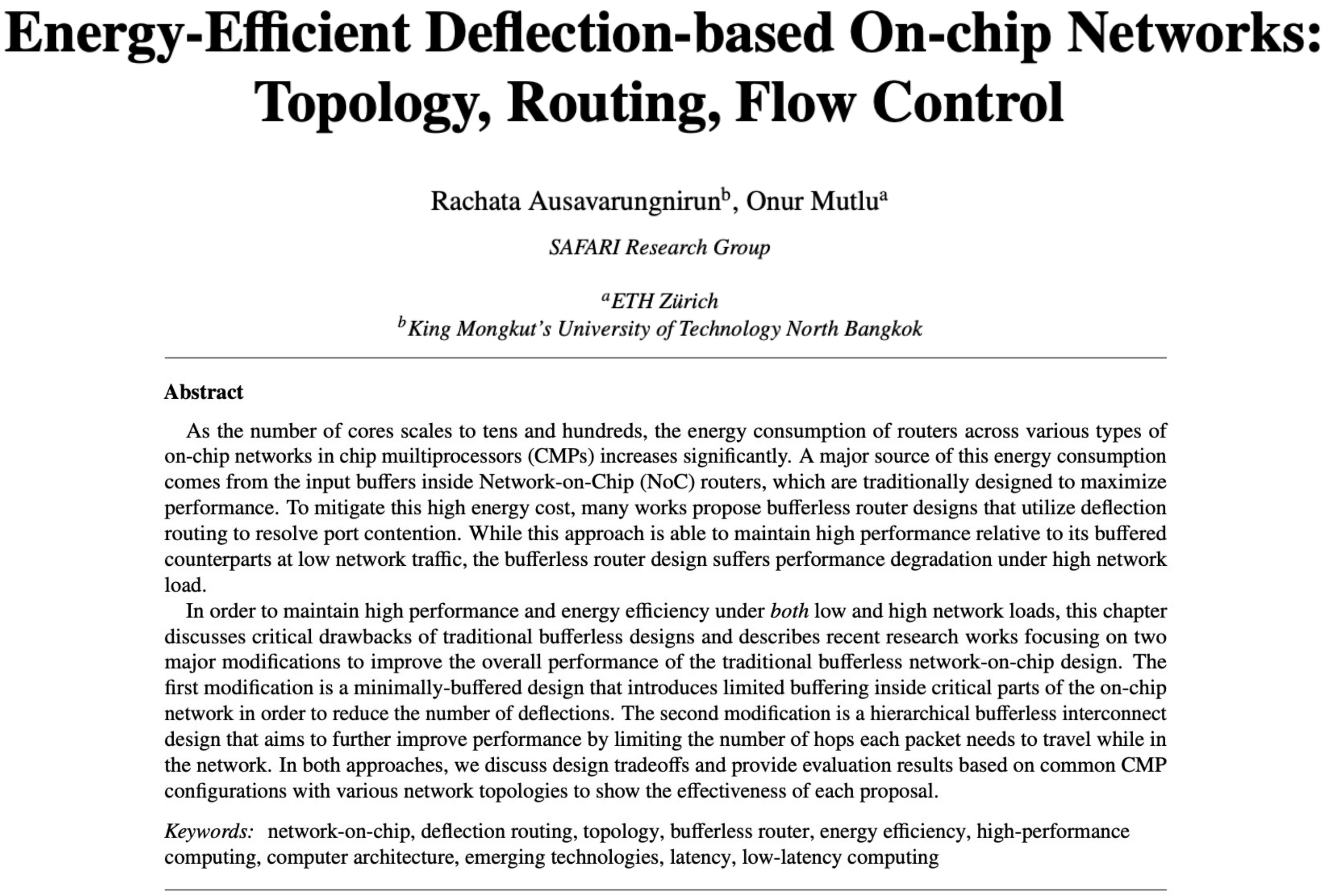
- Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, and Onur Mutlu,
"Bufferless and Minimally-Buffered Deflection Routing"
Invited Book Chapter in Routing Algorithms in Networks-on-Chip, pp. 241-275, Springer, 2014.

Chapter 1

Bufferless and Minimally-Buffered Deflection Routing

Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, Onur Mutlu

Summary of Eight Years of Research



Energy-Efficient Deflection-based On-chip Networks: Topology, Routing, Flow Control

Rachata Ausavarungnirun[b], Onur Mutlu[a]

SAFARI Research Group

[a] *ETH Zürich*
[b] *King Mongkut's University of Technology North Bangkok*

Abstract

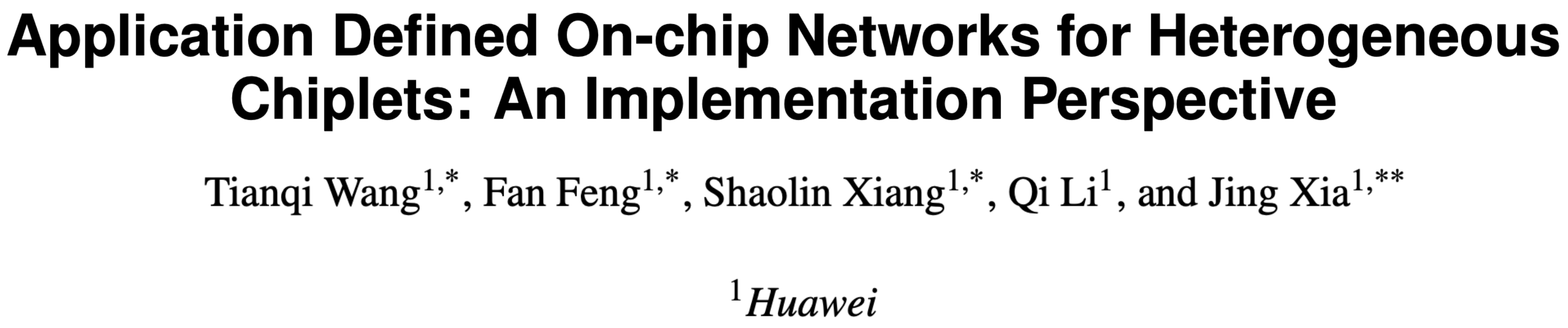
As the number of cores scales to tens and hundreds, the energy consumption of routers across various types of on-chip networks in chip muiltiprocessors (CMPs) increases significantly. A major source of this energy consumption comes from the input buffers inside Network-on-Chip (NoC) routers, which are traditionally designed to maximize performance. To mitigate this high energy cost, many works propose bufferless router designs that utilize deflection routing to resolve port contention. While this approach is able to maintain high performance relative to its buffered counterparts at low network traffic, the bufferless router design suffers performance degradation under high network load.

In order to maintain high performance and energy efficiency under *both* low and high network loads, this chapter discusses critical drawbacks of traditional bufferless designs and describes recent research works focusing on two major modifications to improve the overall performance of the traditional bufferless network-on-chip design. The first modification is a minimally-buffered design that introduces limited buffering inside critical parts of the on-chip network in order to reduce the number of deflections. The second modification is a hierarchical bufferless interconnect design that aims to further improve performance by limiting the number of hops each packet needs to travel while in the network. In both approaches, we discuss design tradeoffs and provide evaluation results based on common CMP configurations with various network topologies to show the effectiveness of each proposal.

Keywords: network-on-chip, deflection routing, topology, bufferless router, energy efficiency, high-performance computing, computer architecture, emerging technologies, latency, low-latency computing

https://arxiv.org/pdf/2112.02516.pdf

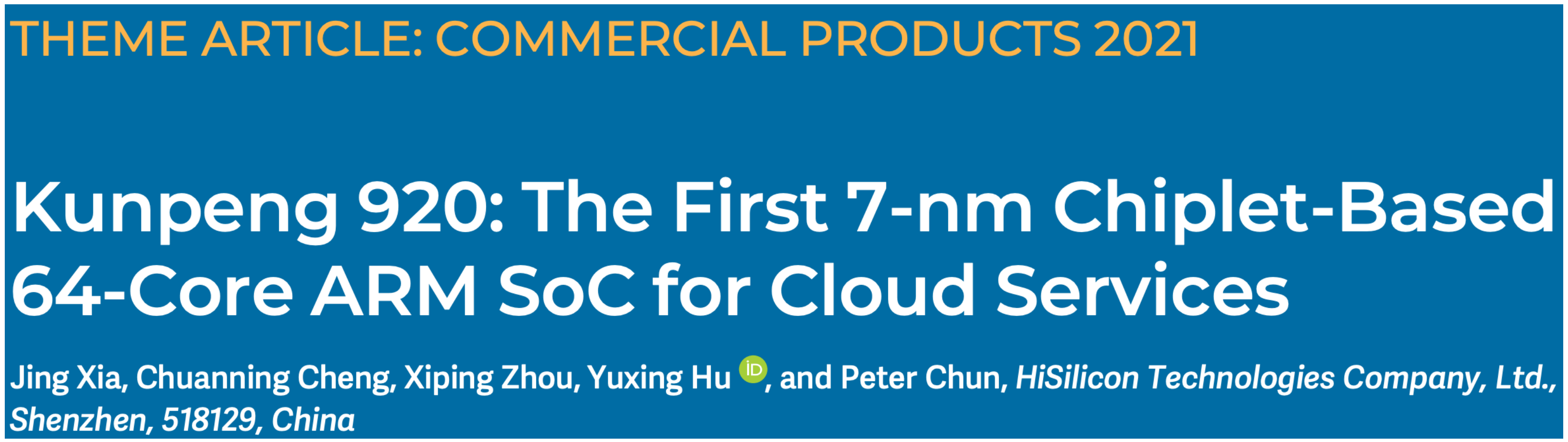
Bufferless Interconnects in Real Systems


Application Defined On-chip Networks for Heterogeneous Chiplets: An Implementation Perspective

Tianqi Wang[1,*], Fan Feng[1,*], Shaolin Xiang[1,*], Qi Li[1], and Jing Xia[1,**]

[1] *Huawei*

https://ieeexplore.ieee.org/document/9773184/

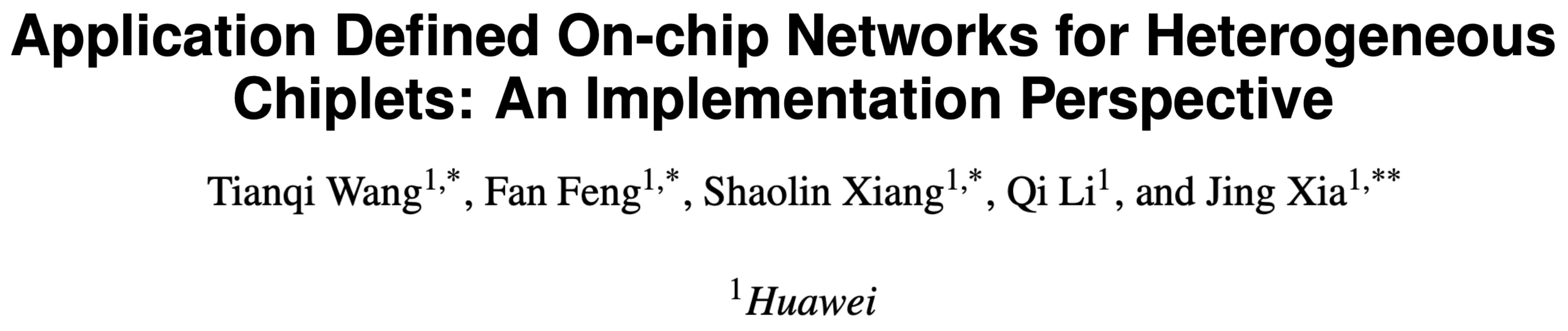

THEME ARTICLE: COMMERCIAL PRODUCTS 2021

Kunpeng 920: The First 7-nm Chiplet-Based 64-Core ARM SoC for Cloud Services

Jing Xia, Chuanning Cheng, Xiping Zhou, Yuxing Hu, and Peter Chun, *HiSilicon Technologies Company, Ltd., Shenzhen, 518129, China*

https://ieeexplore.ieee.org/abstract/document/9444893

Bufferless Interconnects in Real Systems

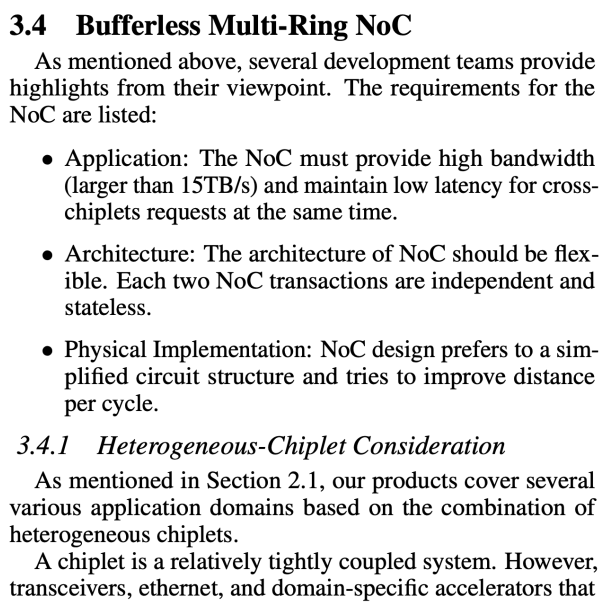


Application Defined On-chip Networks for Heterogeneous Chiplets: An Implementation Perspective

Tianqi Wang[1,*], Fan Feng[1,*], Shaolin Xiang[1,*], Qi Li[1], and Jing Xia[1,**]

[1] *Huawei*

https://ieeexplore.ieee.org/document/9773184/



3.4 Bufferless Multi-Ring NoC

As mentioned above, several development teams provide highlights from their viewpoint. The requirements for the NoC are listed:

- Application: The NoC must provide high bandwidth (larger than 15TB/s) and maintain low latency for cross-chiplets requests at the same time.
- Architecture: The architecture of NoC should be flexible. Each two NoC transactions are independent and stateless.
- Physical Implementation: NoC design prefers to a simplified circuit structure and tries to improve distance per cycle.

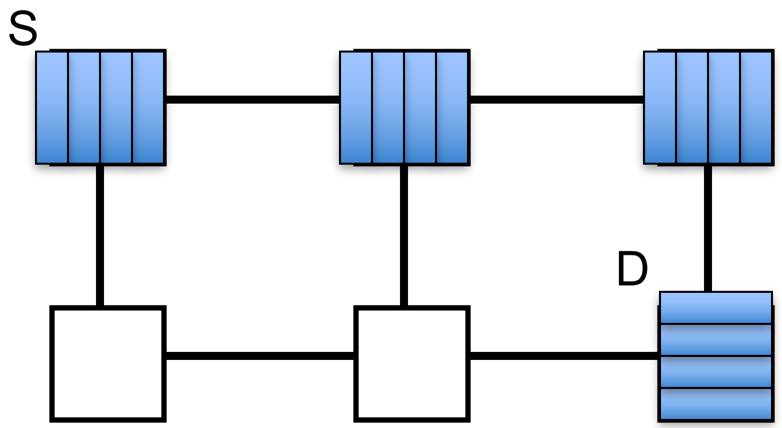
3.4.1 Heterogeneous-Chiplet Consideration

As mentioned in Section 2.1, our products cover several various application domains based on the combination of heterogeneous chiplets.

A chiplet is a relatively tightly coupled system. However, transceivers, ethernet, and domain-specific accelerators that

Store and Forward Flow Control

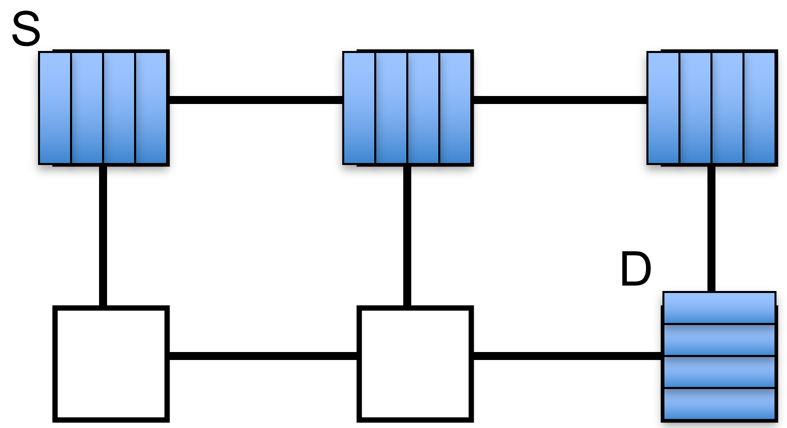
- Packet-based flow control
- Store and Forward
  - Packet copied entirely into network router before moving to the next node
  - Flow control unit is the entire packet
- Leads to high per-packet latency
- Requires buffering for entire packet in each node



Can we do better?

Cut-Through Flow Control

- Another form of packet-based flow control
- Start forwarding as soon as header is received and resources (buffer, channel, etc) allocated
  - Large reduction in latency
- Still allocates buffers and channel bandwidth for full packets



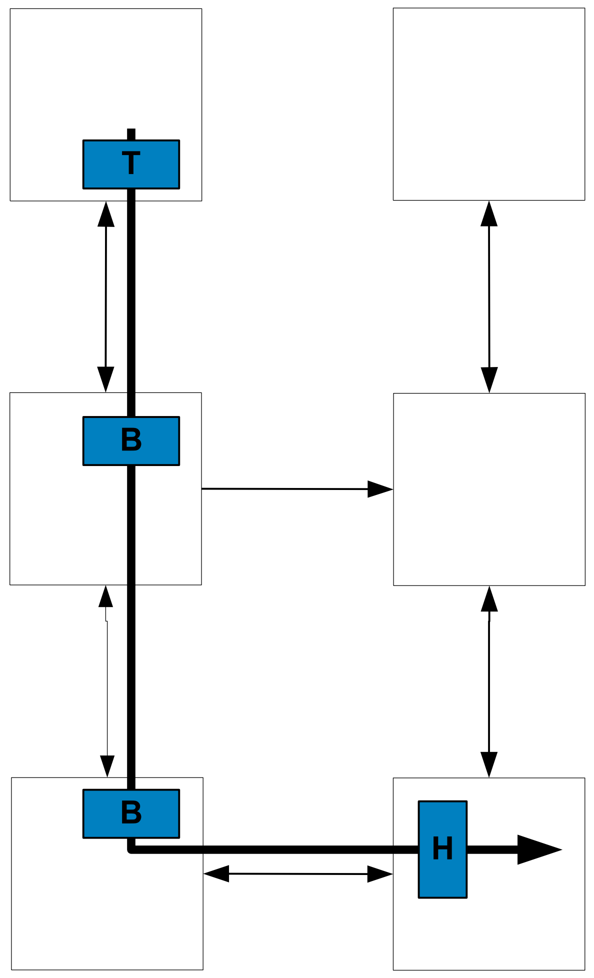
- What if packets are large?

Cut-Through Flow Control

- What to do if output port is blocked?
- Lets the tail continue when the head is blocked, absorbing the whole message into a single switch.
  - Requires a buffer large enough to hold the largest packet.
- Degenerates to store-and-forward with high contention

- **Can we do better?**

Wormhole Flow Control



- Packets broken into (potentially) smaller flits (buffer/bw allocation unit)
- 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
- How does body/tail know where to go?
  - Follow the head (need state in router)
- Latency almost independent of distance for long messages

Wormhole Flow Control

- Advantages over “store and forward” flow control

  + Lower latency

  + More efficient buffer utilization

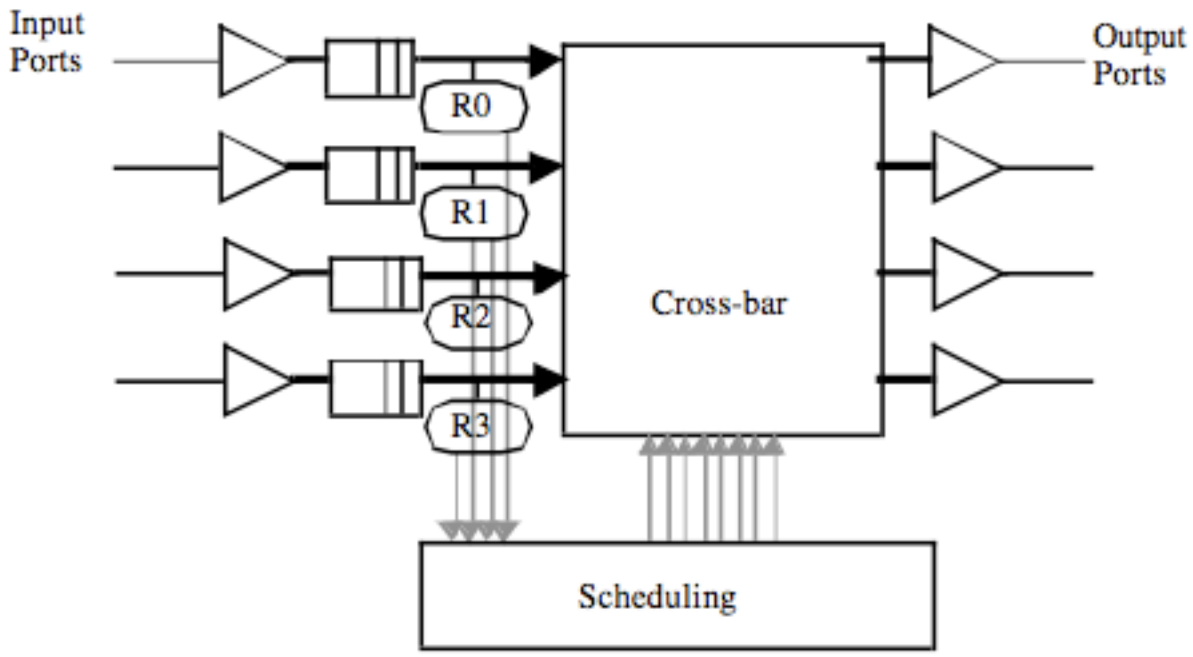
- Limitations

  - Suffers from **head of line blocking**

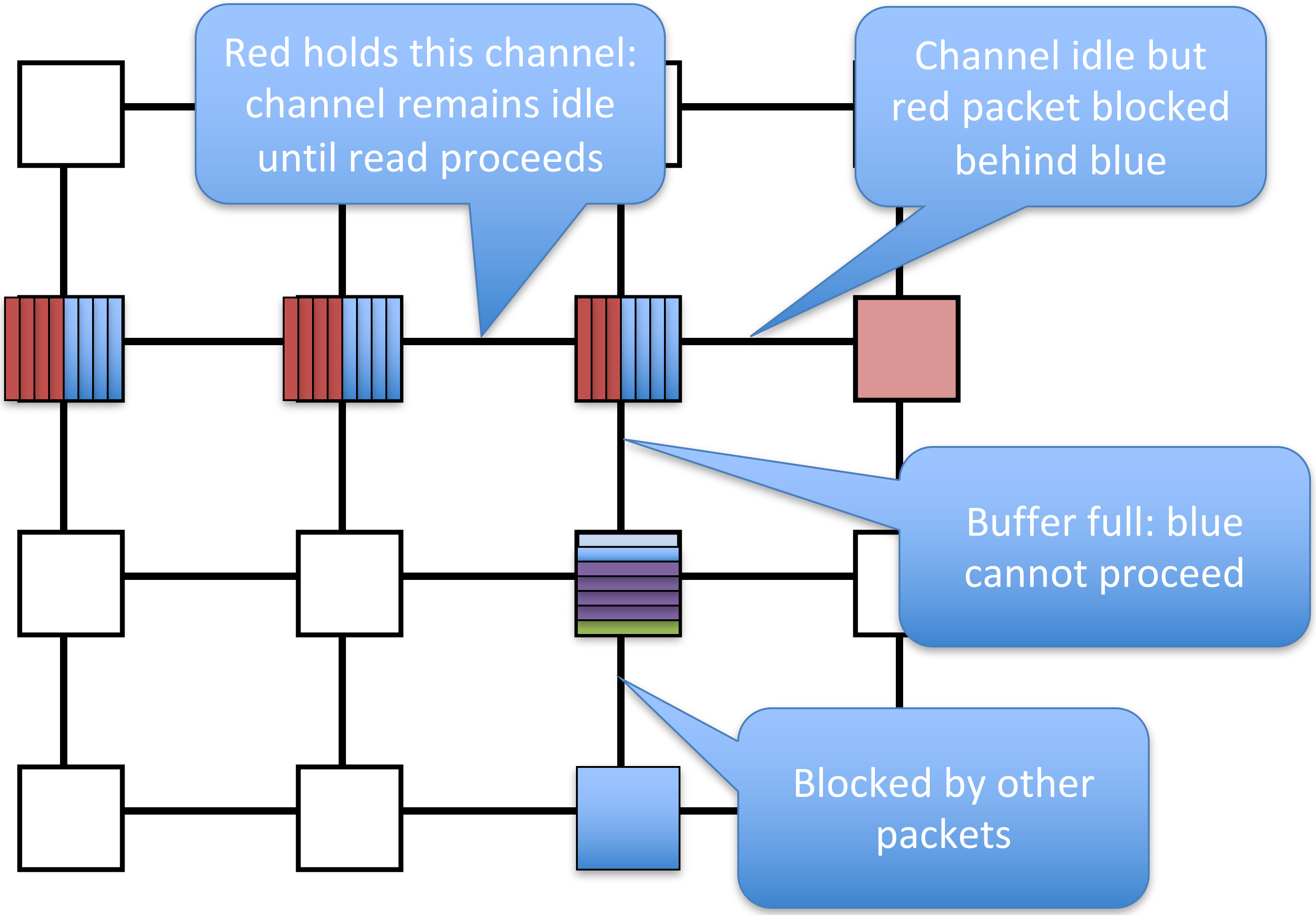
  - If head flit cannot move due to contention, another worm cannot proceed even though links may be idle

Head of Line Blocking

- A worm can be before another in the router input buffer
- Due to FIFO nature, the second worm cannot be scheduled even though it may need to access another output port

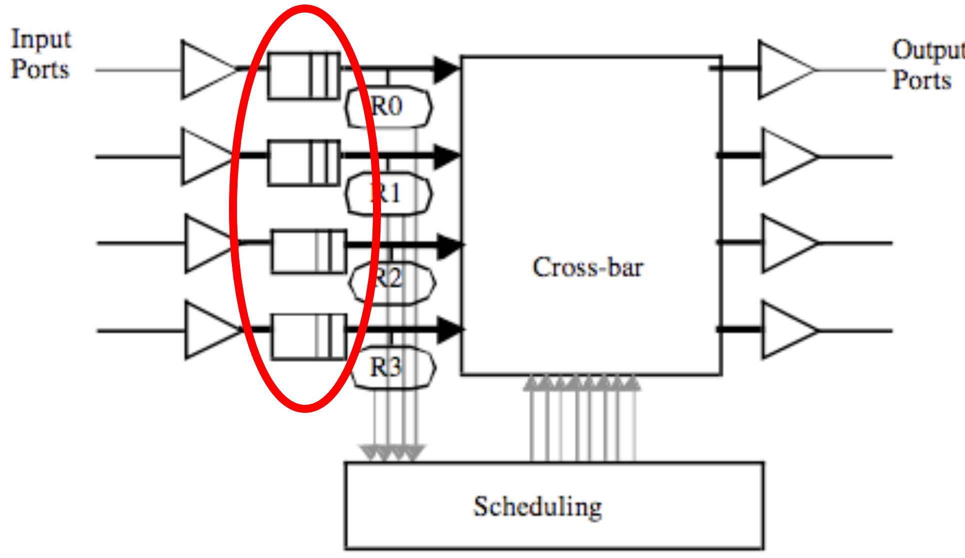


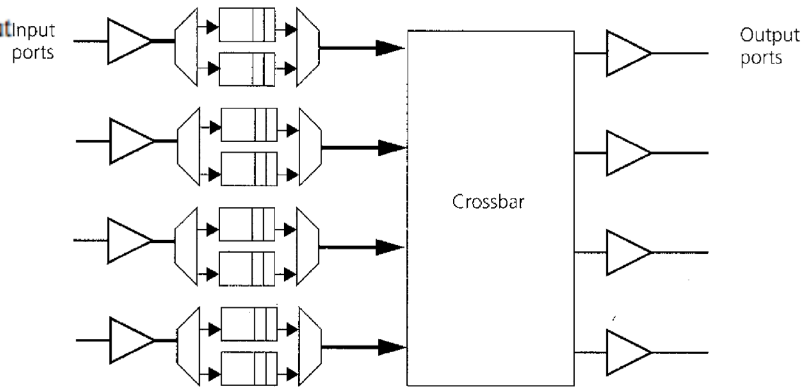
Head of Line Blocking


Red holds this channel: channel remains idle until read proceeds
Channel idle but red packet blocked behind blue
Buffer full: blue cannot proceed
Blocked by other packets

Virtual Channel Flow Control

- Idea: Multiplex multiple channels over one physical channel
- Divide up the input buffer into multiple buffers sharing a single physical channel
- Dally, "Virtual Channel Flow Control," ISCA 1990.





Virtual Channel Flow Control

- Idea: Multiplex multiple channels over one physical channel
- Divide up the input buffer into multiple buffers sharing a single physical channel
- Dally, "Virtual Channel Flow Control," ISCA 1990.

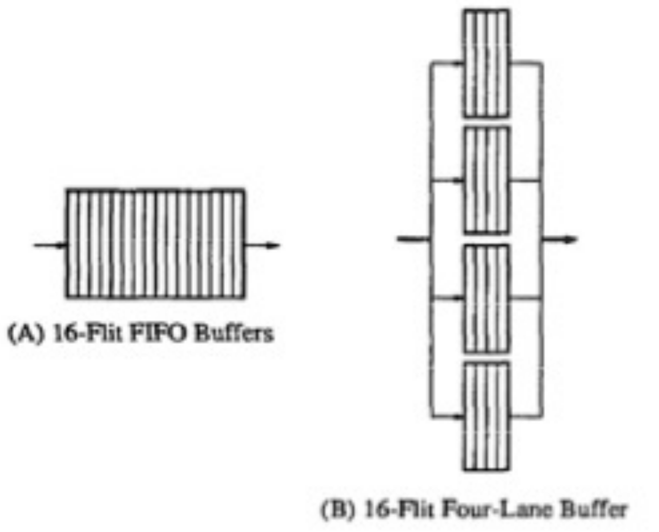
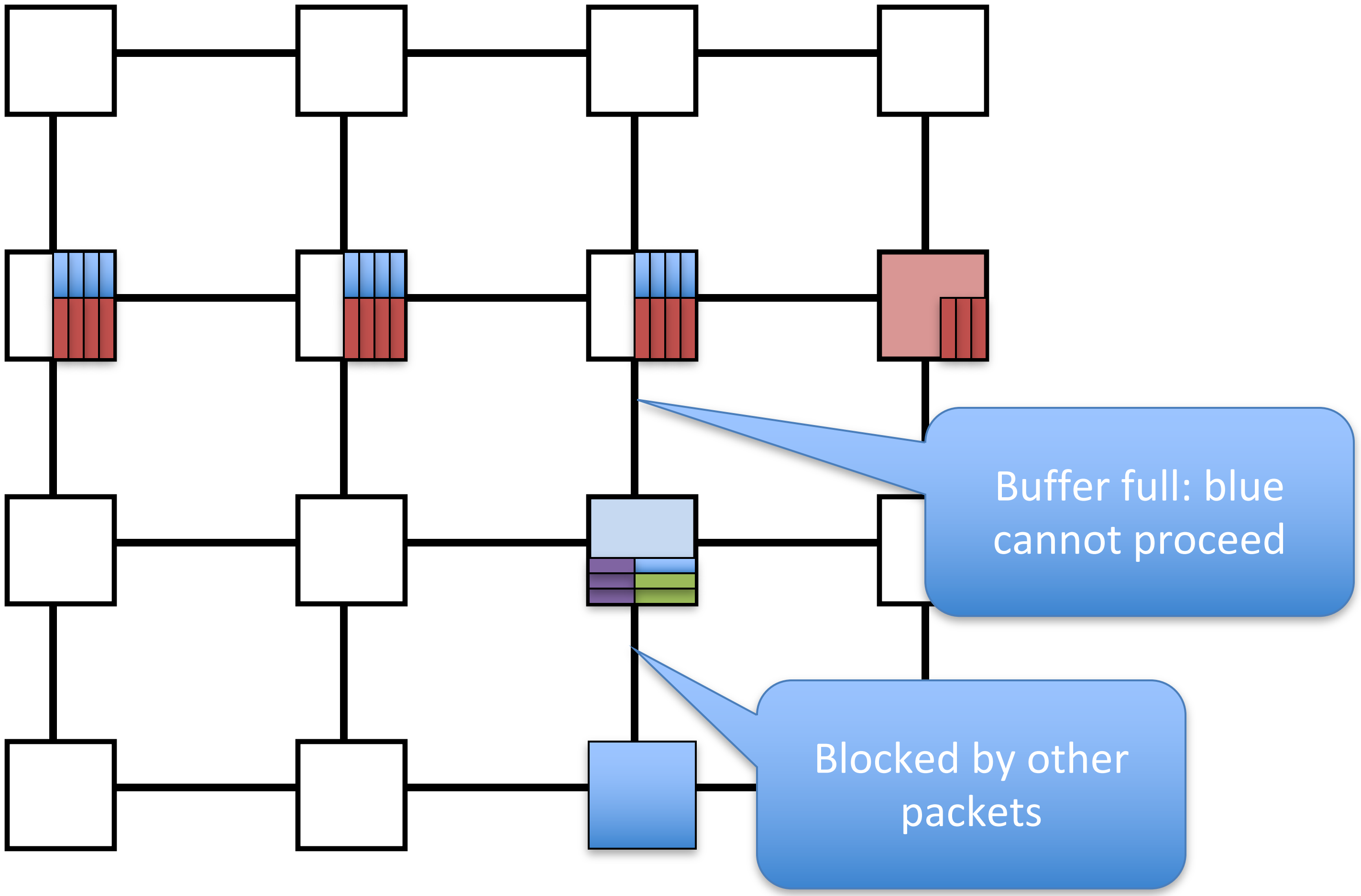
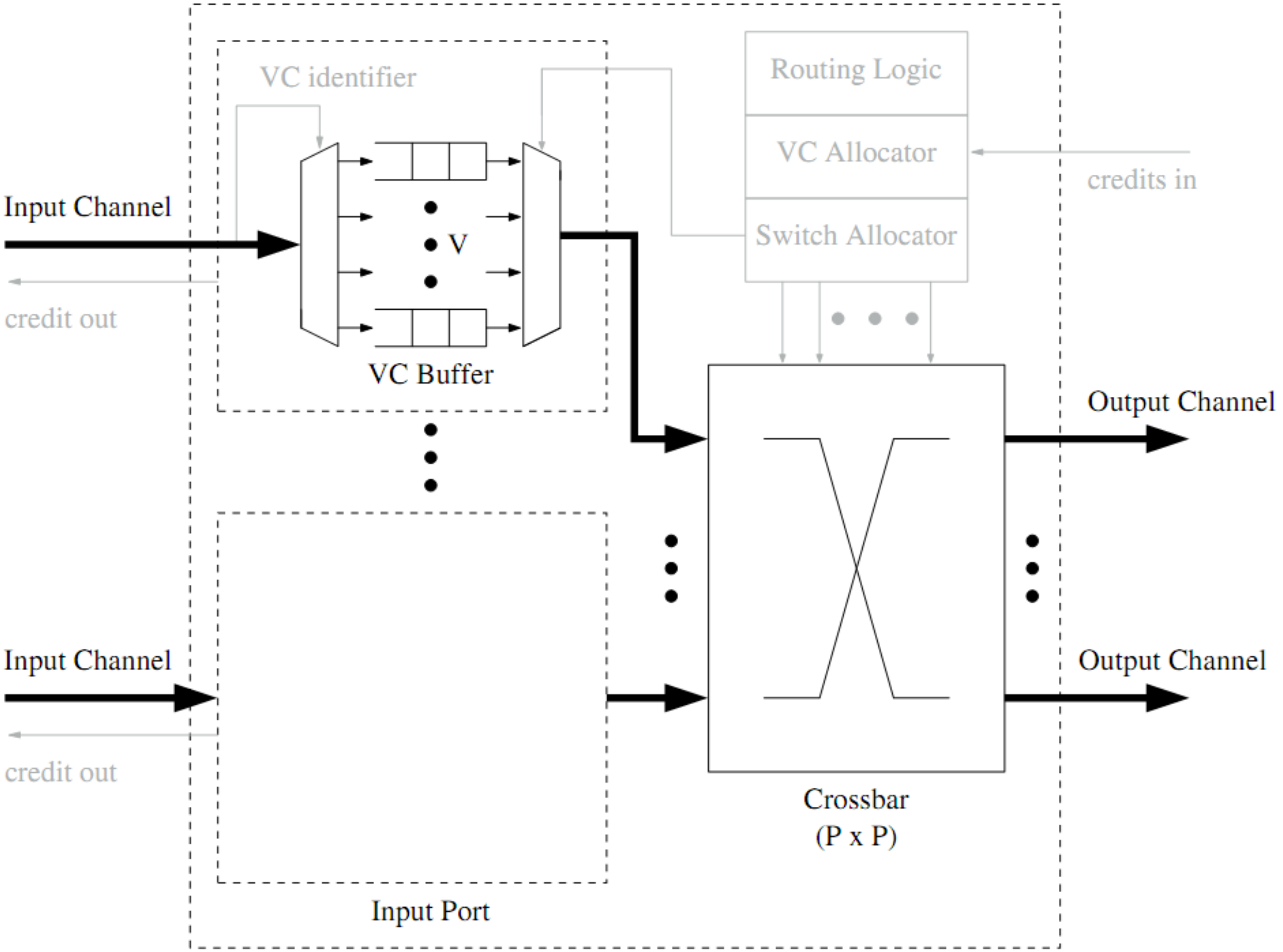


Figure 5: (A) Conventional nodes organize their buffers into FIFO queues restricting routing. (B) A network using virtual-channel flow control organizes its buffers into several independent lanes.

Virtual Channel Flow Control



A Modern Virtual Channel Based Router

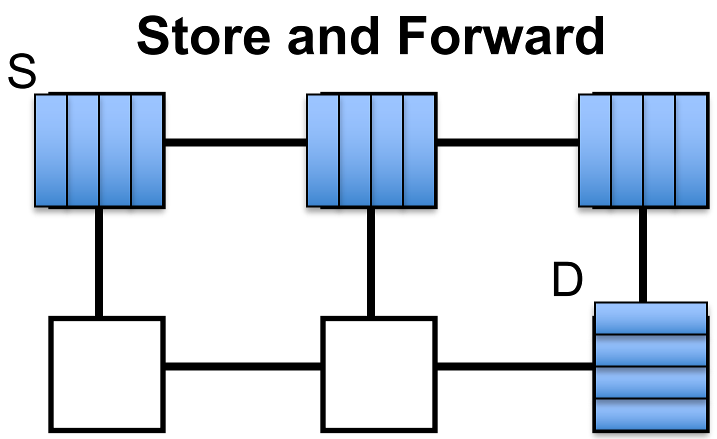


Other Uses of Virtual Channels

- Deadlock avoidance
  - Enforcing switching to a different set of virtual channels on some "turns" can break the cyclic dependency of resources
    - Enforce order on VCs
  - Escape VCs: Have at least one VC that uses deadlock-free routing. Ensure each flit has fair access to that VC.
  - Protocol level deadlock: Ensure address and data packets use different VCs → prevent cycles due to intermixing of different packet classes

- Prioritization of traffic classes
  - Some virtual channels can have higher priority than others

Review: Flow Control

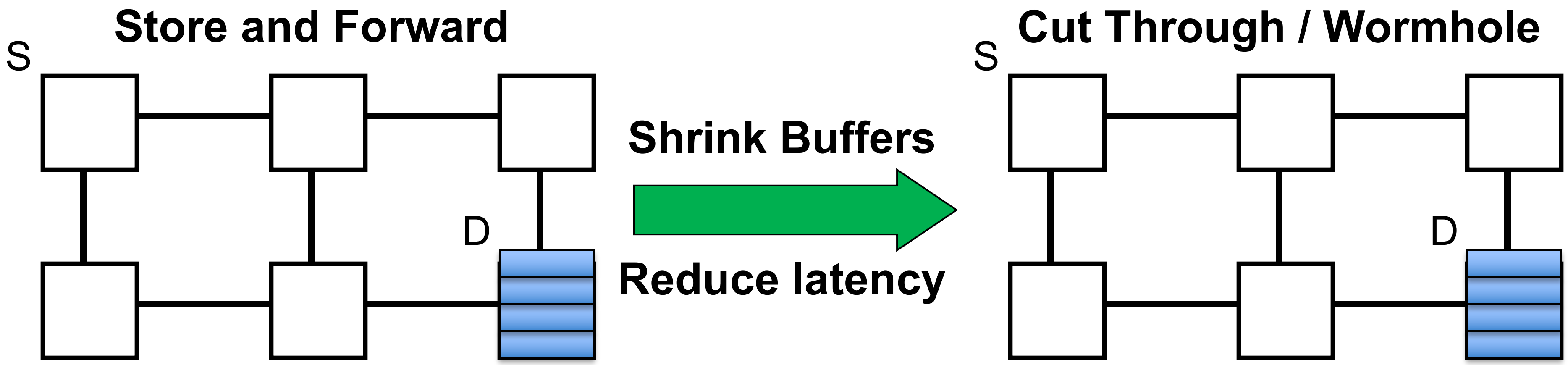


Any other issues?

Head-of-Line Blocking

Use Virtual Channels

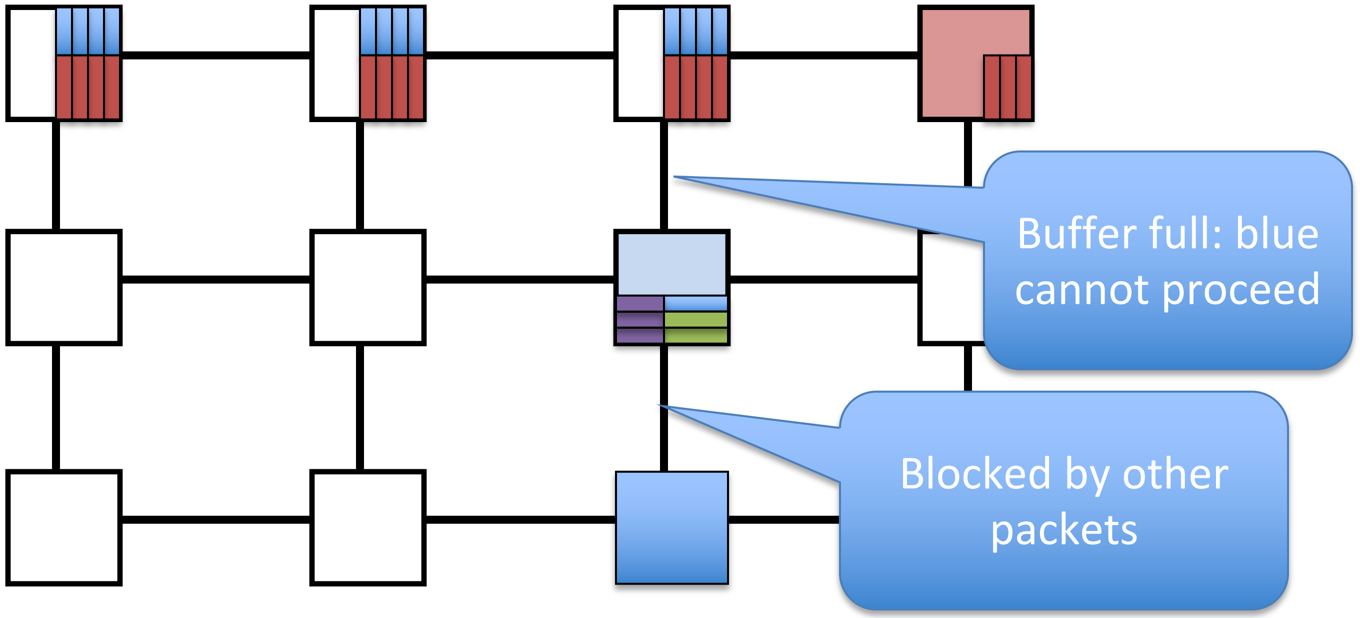
Review: Flow Control



Any other issues?

Head-of-Line Blocking

Use Virtual Channels



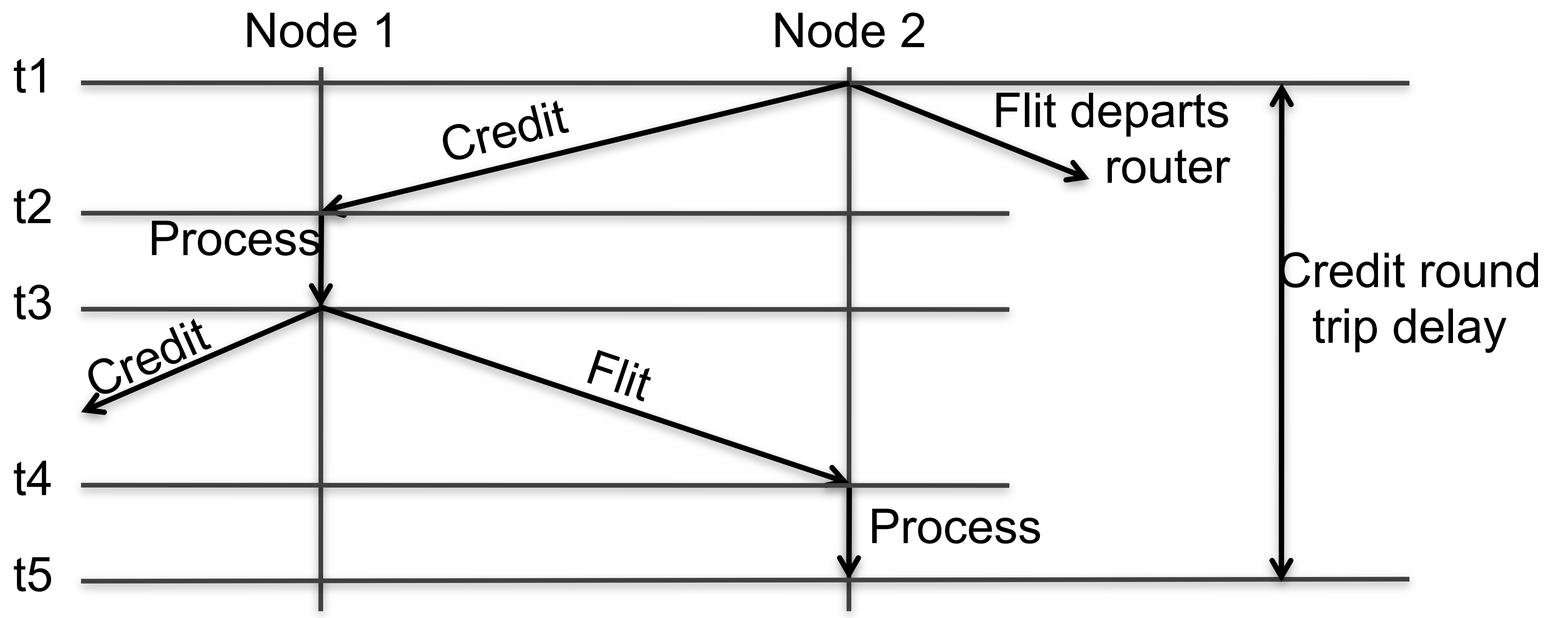
Communicating Buffer Availability

- Credit-based flow control
  - Upstream knows how many buffers are downstream
  - Downstream passes back credits to upstream
  - Significant upstream signaling (esp. for small flits)

- On/Off (XON/XOFF) flow control
  - Downstream has on/off signal to upstream

- ACK/NACK flow control
  - Upstream optimistically sends downstream
  - Buffer cannot be deallocated until ACK/NACK received
  - Inefficiently utilizes buffer space

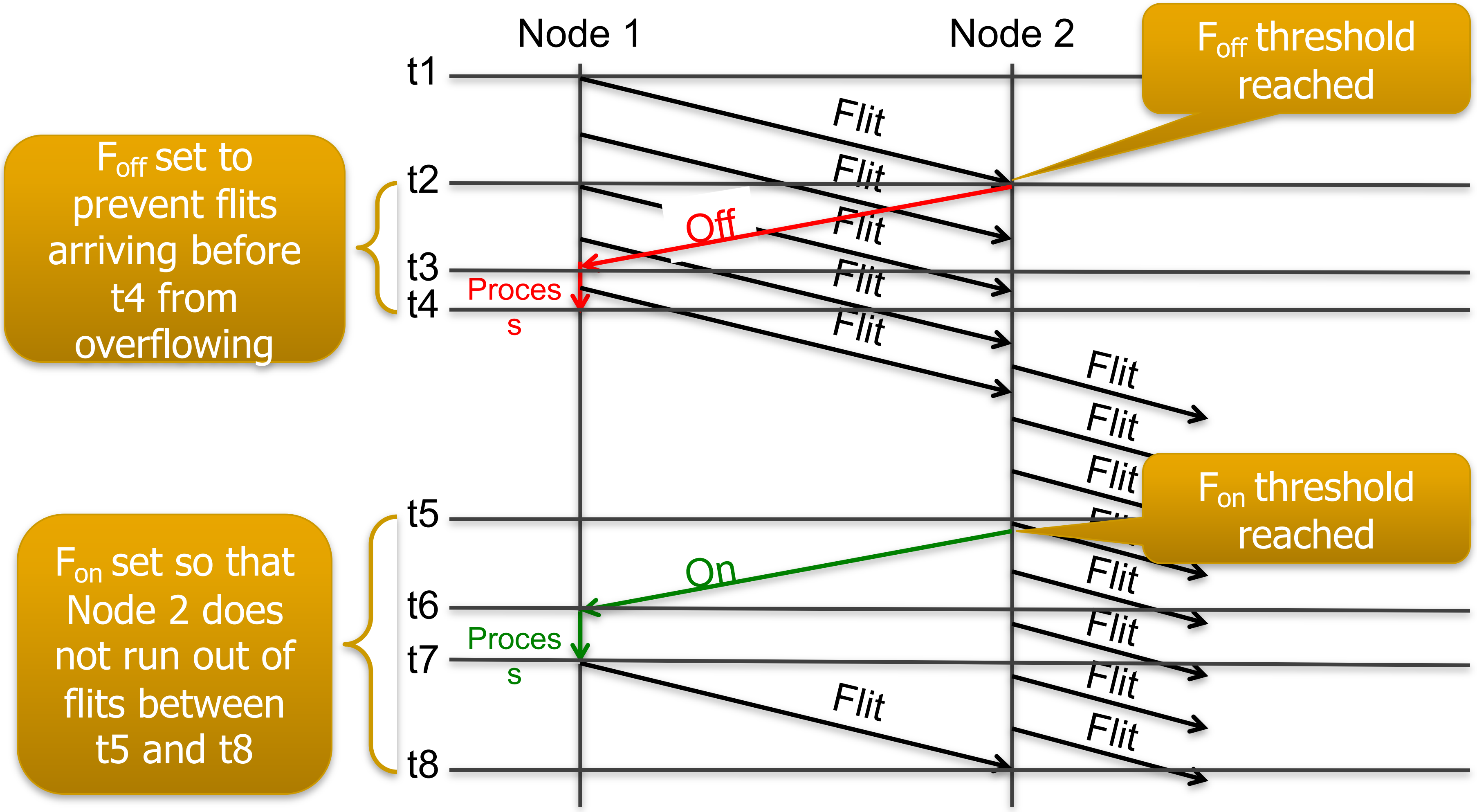
Credit-based Flow Control



- Round-trip credit delay:
  - Time between when buffer empties and when next flit can be processed from that buffer entry
- Significant throughput degradation if there are few buffers
- Important to size buffers to tolerate credit turn-around

On/Off (XON/XOFF) Flow Control

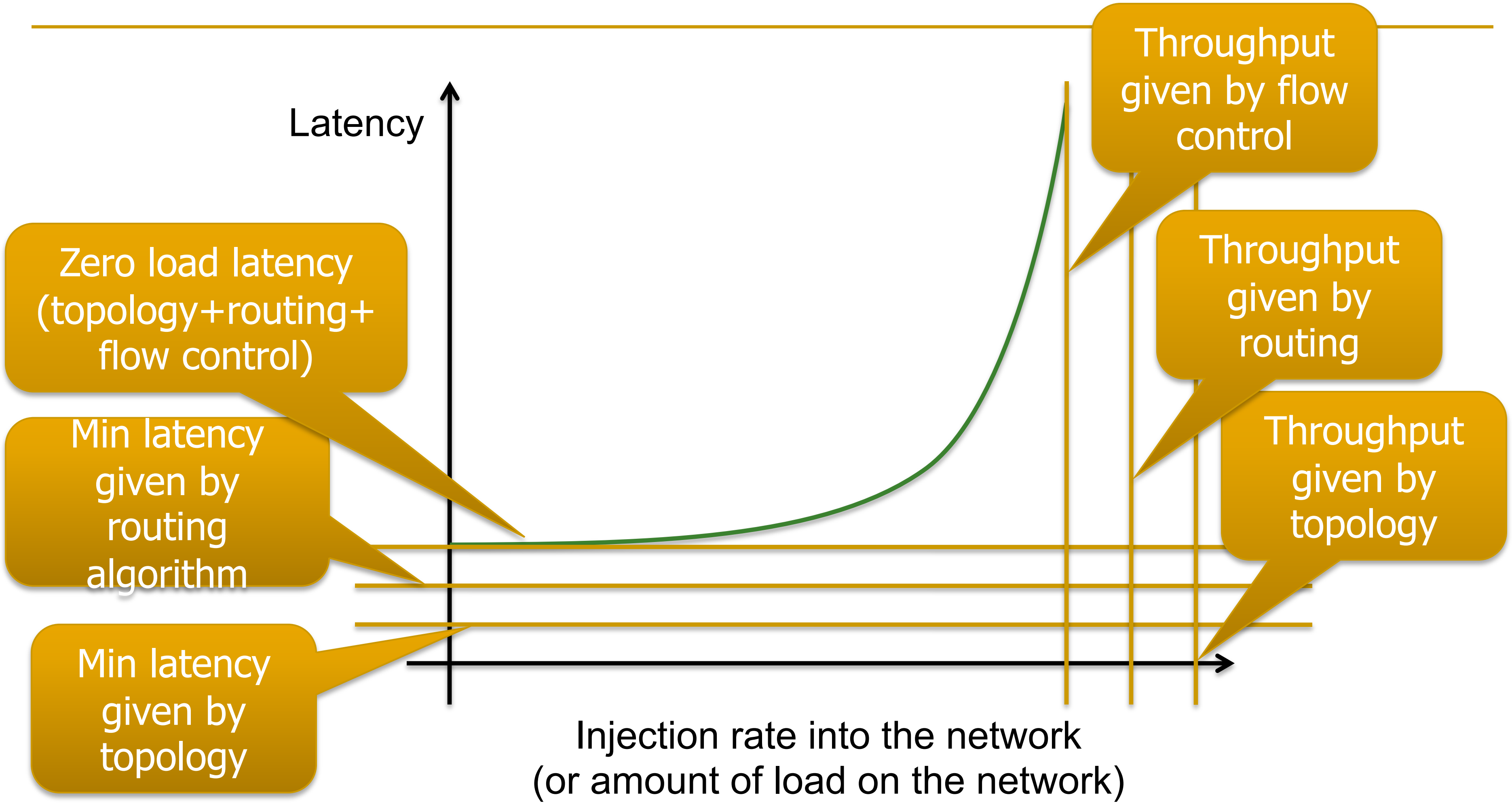
- Downstream has on/off signal to upstream



Node 1
Node 2
t1
t2
t3
t4
t5
t6
t7
t8
Flit
Off
On
Process
F_{off} threshold reached
F_{on} threshold reached
F_{off} set to prevent flits arriving before t4 from overflowing
F_{on} set so that Node 2 does not run out of flits between t5 and t8

Interconnection Network Performance

Interconnection Network Performance


Latency
Injection rate into the network (or amount of load on the network)
Zero load latency (topology+routing+ flow control)
Min latency given by routing algorithm
Min latency given by topology
Throughput given by flow control
Throughput given by routing
Throughput given by topology

Saturation throughput: Injection rate at which latency asymptotes

Ideal Latency

- Ideal latency
  - Solely due to wire delay between source and destination

$$T_{ideal} = \frac{D}{v} + \frac{L}{b}$$

  - D = Manhattan distance
    - The distance between two points measured along axes at right angles.
  - v = propagation velocity
  - L = packet size
  - b = channel bandwidth

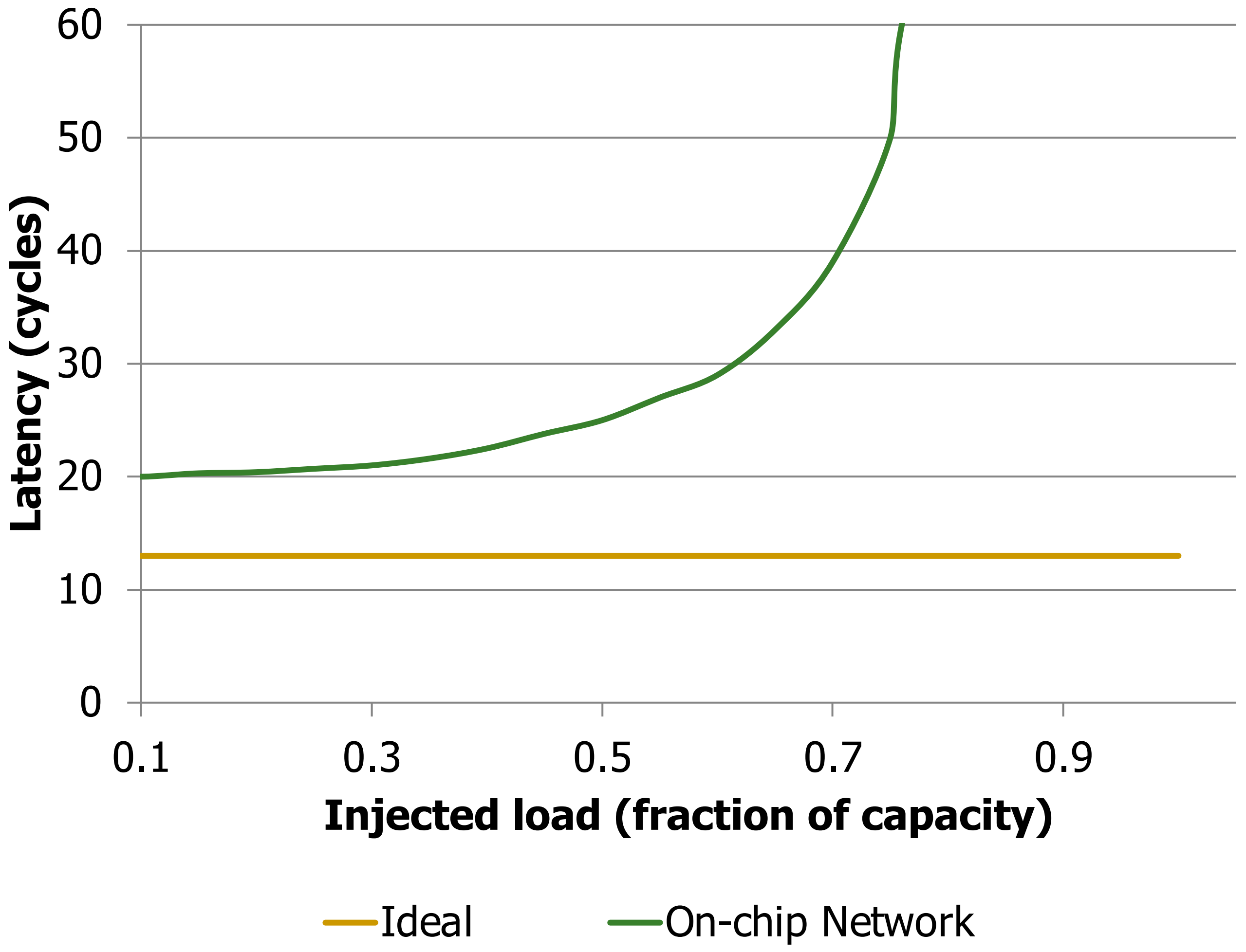
Actual Latency

- Dedicated wiring impractical
  - Long wires segmented with insertion of routers

$$T_{actual} = \frac{D}{v} + \frac{L}{b} + H \cdot T_{router} + T_c$$

  - D = Manhattan distance
  - v = propagation velocity
  - L = packet size
  - b = channel bandwidth
  - H = hops
  - T_{router} = router latency
  - T_c = latency due to contention

Load-Latency Curve



Load-Latency Curve Examples

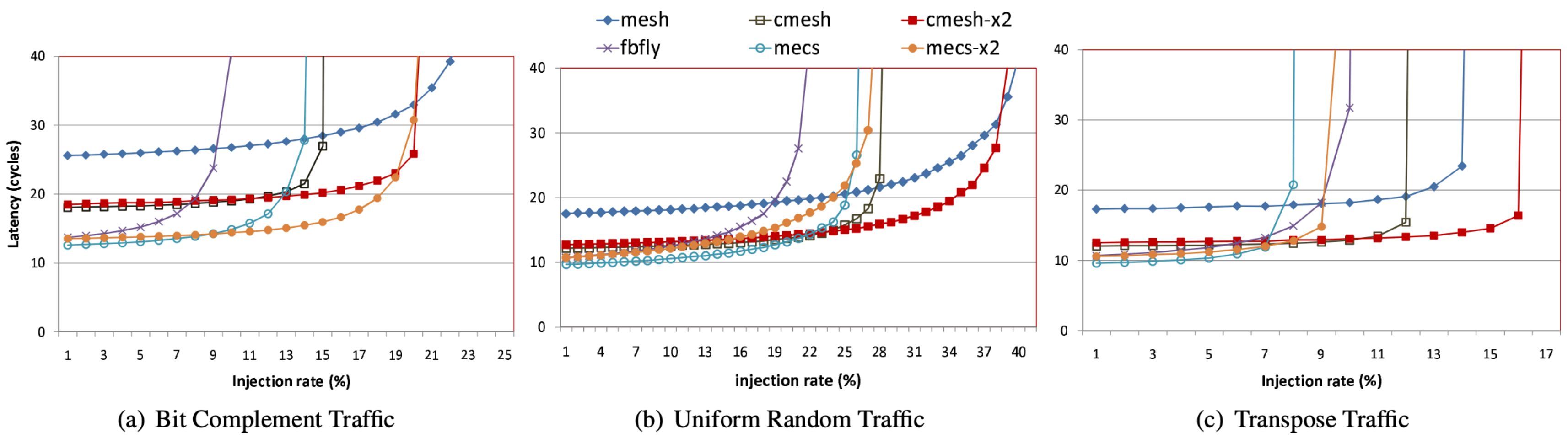


Figure 4. Load-latency graphs for 64-node mesh, CMesh, flattened butterfly and MECS topologies.

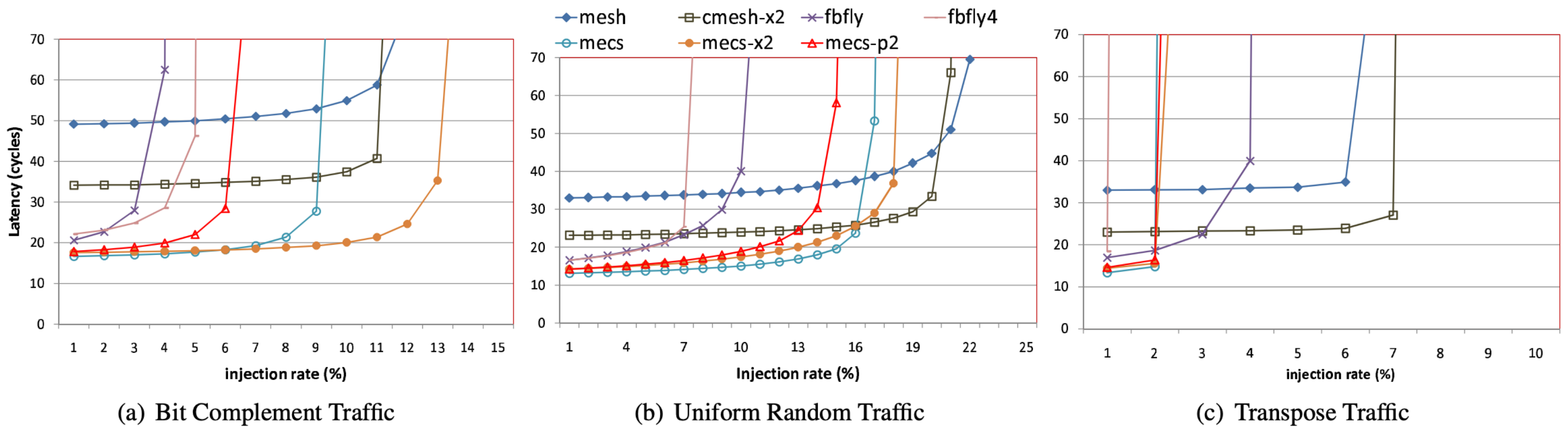
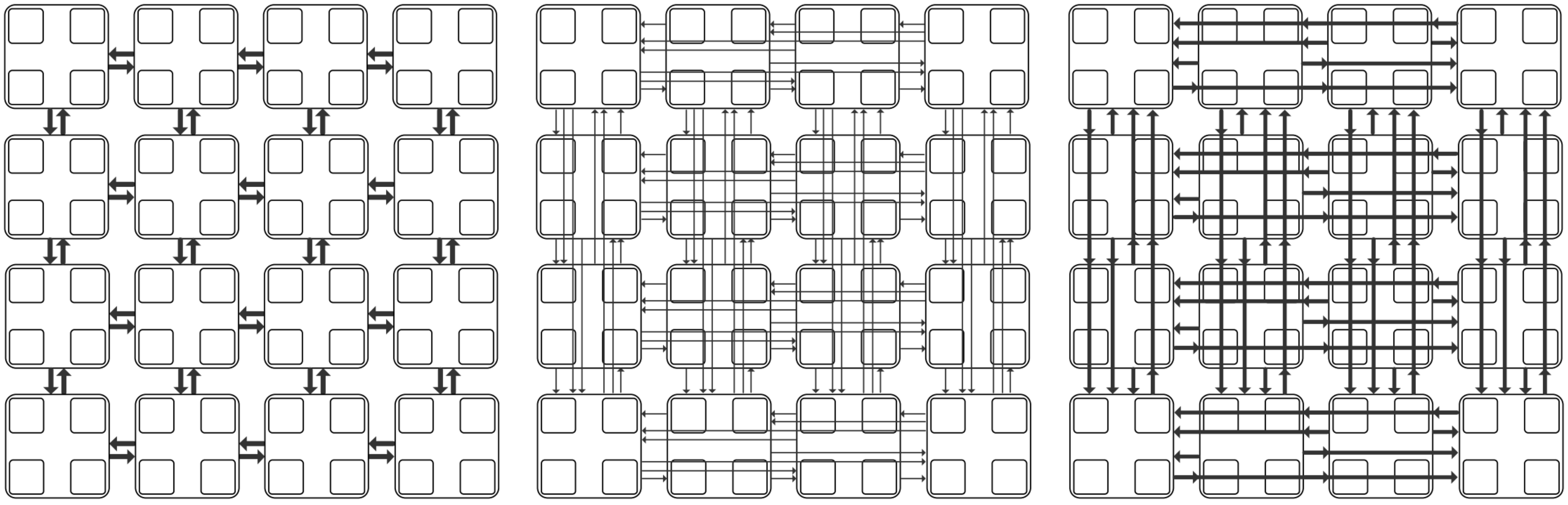


Figure 5. Load-latency graphs for 256-node mesh, CMesh, flattened butterfly and MECS topologies.

Grot+, "Express Cube Topologies for On-Chip Interconnects," HPCA 2009.

Examined Topologies in Prior Slide



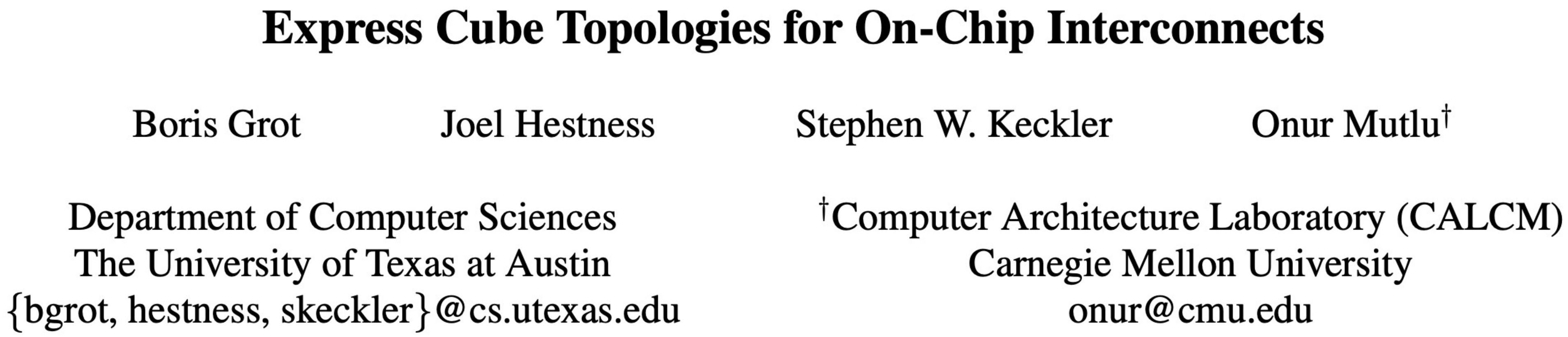
(a) Concentrated mesh (b) Flattened butterfly (c) MECS

Figure 1. Concentrated Mesh, Flattened Butterfly and MECS topologies for a 64-terminal network.

Grot+, "Express Cube Topologies for On-Chip Interconnects," HPCA 2009. 117

Multi-Drop Express Channels (MECS)

- Boris Grot, Joel Hestness, Stephen W. Keckler, and Onur Mutlu, **"Express Cube Topologies for On-Chip Interconnects"** *Proceedings of the 15th International Symposium on High-Performance Computer Architecture* (***HPCA***), pages 163-174, Raleigh, NC, February 2009. Slides (ppt)



Express Cube Topologies for On-Chip Interconnects

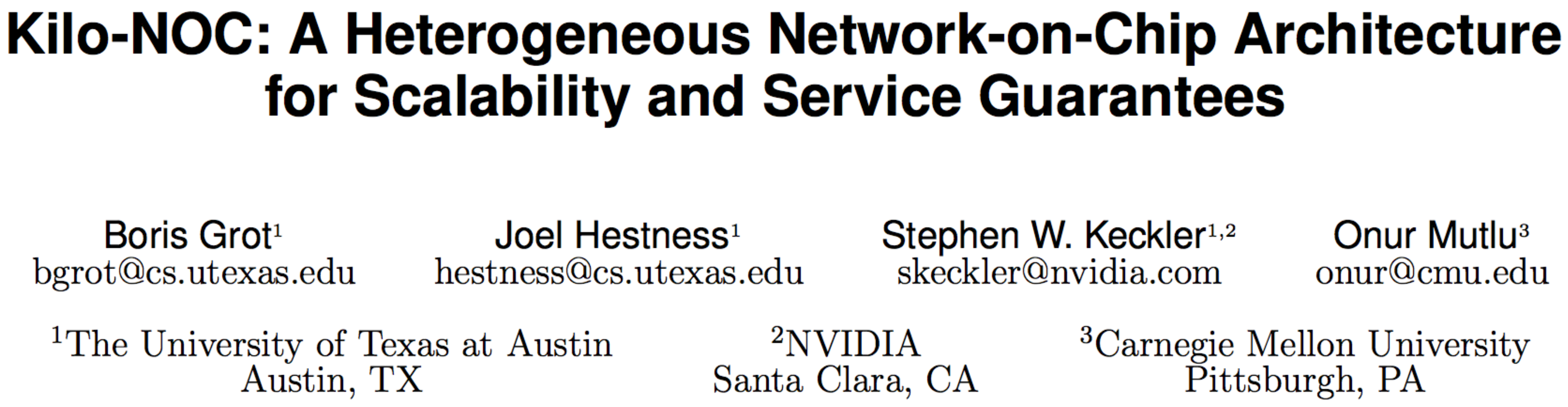
Boris Grot Joel Hestness Stephen W. Keckler Onur Mutlu[†]

Department of Computer Sciences
The University of Texas at Austin
{bgrot, hestness, skeckler}@cs.utexas.edu

[†]Computer Architecture Laboratory (CALCM)
Carnegie Mellon University
onur@cmu.edu

Kilo-NoC Building on MECS

- Boris Grot, Joel Hestness, Stephen W. Keckler, and Onur Mutlu, **"Kilo-NOC: A Heterogeneous Network-on-Chip Architecture for Scalability and Service Guarantees"**
Proceedings of the 38th International Symposium on Computer Architecture (***ISCA***), San Jose, CA, June 2011. Slides (pptx)
One of the 12 computer architecture papers of 2011 selected as Top Picks by IEEE Micro.



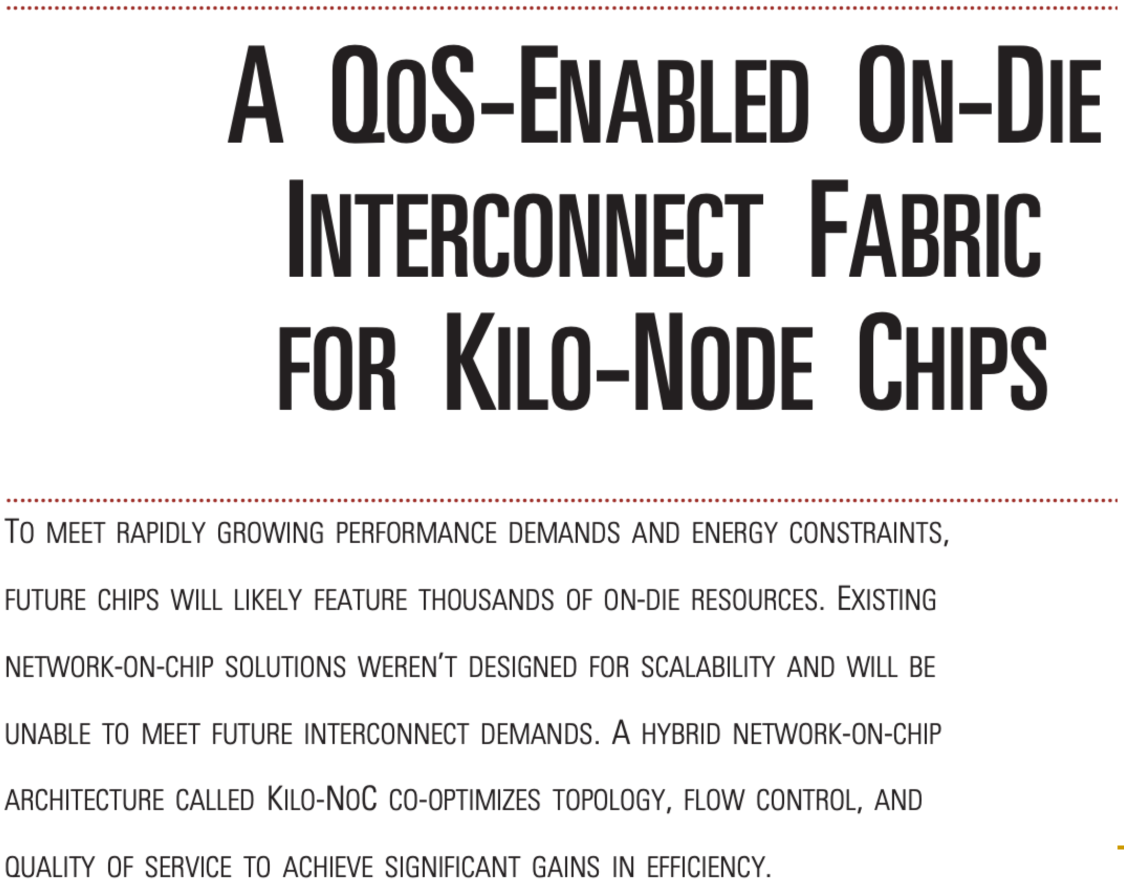
Kilo-NOC: A Heterogeneous Network-on-Chip Architecture for Scalability and Service Guarantees

Boris Grot[1] bgrot@cs.utexas.edu
Joel Hestness[1] hestness@cs.utexas.edu
Stephen W. Keckler[1,2] skeckler@nvidia.com
Onur Mutlu[3] onur@cmu.edu

[1]The University of Texas at Austin, Austin, TX
[2]NVIDIA, Santa Clara, CA
[3]Carnegie Mellon University, Pittsburgh, PA

Kilo-NoC Building on MECS

- Boris Grot, Joel Hestness, Stephen W. Keckler, and Onur Mutlu, **"A QoS-Enabled On-Die Interconnect Fabric for Kilo-Node Chips"** *IEEE Micro, Special Issue: Micro's Top Picks from 2011 Computer Architecture Conferences (**MICRO TOP PICKS**)*, Vol. 32, No. 3, May/June 2012.



A QOS-ENABLED ON-DIE INTERCONNECT FABRIC FOR KILO-NODE CHIPS

TO MEET RAPIDLY GROWING PERFORMANCE DEMANDS AND ENERGY CONSTRAINTS, FUTURE CHIPS WILL LIKELY FEATURE THOUSANDS OF ON-DIE RESOURCES. EXISTING NETWORK-ON-CHIP SOLUTIONS WEREN'T DESIGNED FOR SCALABILITY AND WILL BE UNABLE TO MEET FUTURE INTERCONNECT DEMANDS. A HYBRID NETWORK-ON-CHIP ARCHITECTURE CALLED KILO-NOC CO-OPTIMIZES TOPOLOGY, FLOW CONTROL, AND QUALITY OF SERVICE TO ACHIEVE SIGNIFICANT GAINS IN EFFICIENCY.

Network Performance Metrics

- Packet latency (avg/max)

- Round trip latency (avg/max)

- Saturation throughput

- Application-level performance: execution time
- System performance: job throughput
  - Affected by interference among threads/applications

Computer Architecture
Lecture 20: Interconnects

Prof. Onur Mutlu
ETH Zürich

Fall 2022

2 December 2022