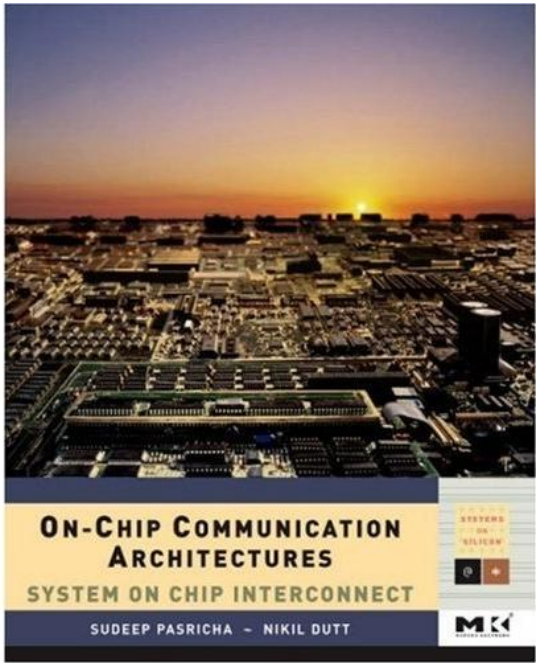
Networks-on-Chip (NoC)



On-Chip Communication Architectures

Some of the lecture notes are based on slides of the course ECE/CS 757 University of Wisconsin-Madison, Spring 2013 by InstructorMikko H Lipasti

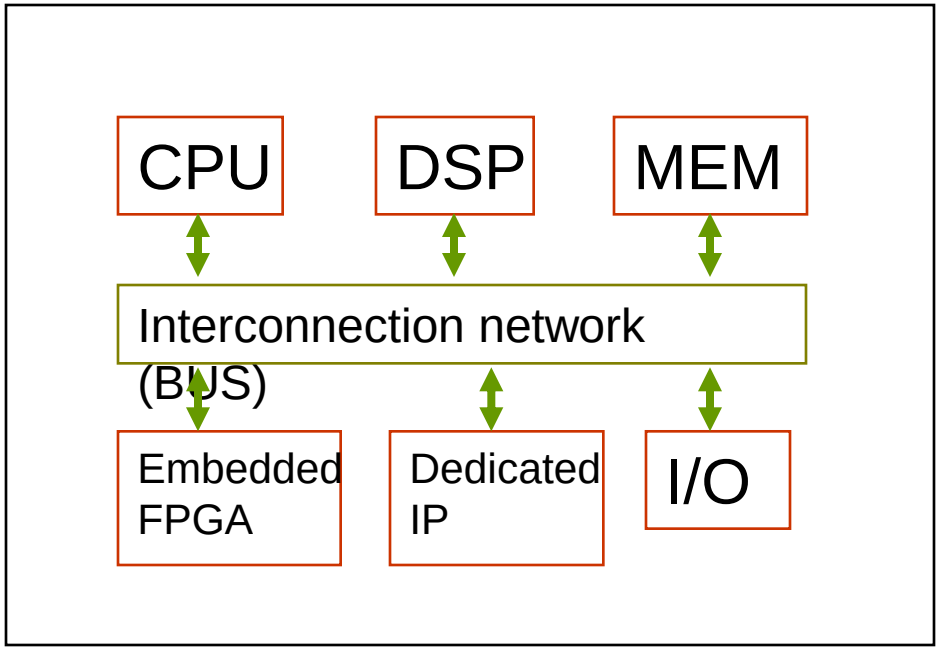
Outline

- Introduction
- NoC Topology
- Switching strategies
- Routing algorithms
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- Status and Open Problems

Introduction

- How to connect individual devices into a group of communicating devices?
  - A device can be:
    - Component within a chip
    - Component within a computer
    - Computer
    - System of computers
  - Network consists of:
    - End point devices with interface to network
    - Links
    - Interconnect hardware
- Goal: transfer maximum amount of information with the least cost (minimum time, power)

Heterogeneous Today's SoC



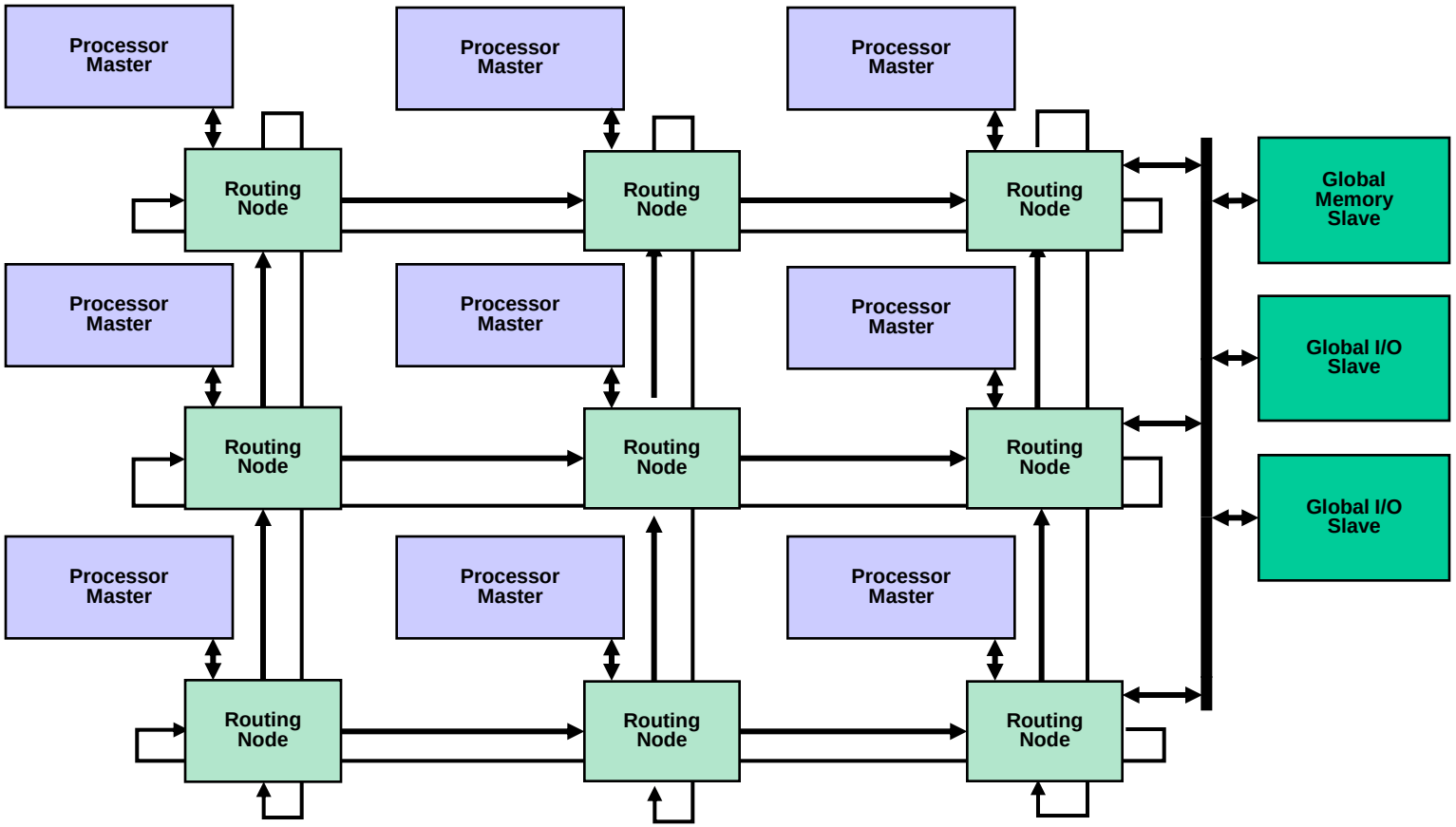
Bus pros (☺) and cons (☹)

- ☹ Every unit attached adds parasitic capacitance, therefore electrical performance degrades with growth.
- ☹ Bus timing is difficult in a deep submicron process.
- ☹ Bus arbiter delay grows with the number of masters. The arbiter is also instance-specific.
- ☹ Bandwidth is limited and shared by all units attached.
- ☺ The silicon cost of a bus is small.
- ☺ Any bus is almost directly compatible with most available IPs, including software running on CPUs.
- ☺ The concepts are simple and well understood.

What are NoC's?

- According to Wikipedia:
  - *"Network-on-a-chip (NoC) is a new paradigm for System-on-Chip (SoC) design. NoC based-systems accommodate multiple asynchronous clocking that many of today's complex SoC designs use. The NoC solution brings a networking method to on-chip communications and claims roughly a threefold performance increase over conventional bus systems."*

NoC exemple


Processor Master
Routing Node
Global Memory Slave
Global I/O Slave
Global I/O Slave

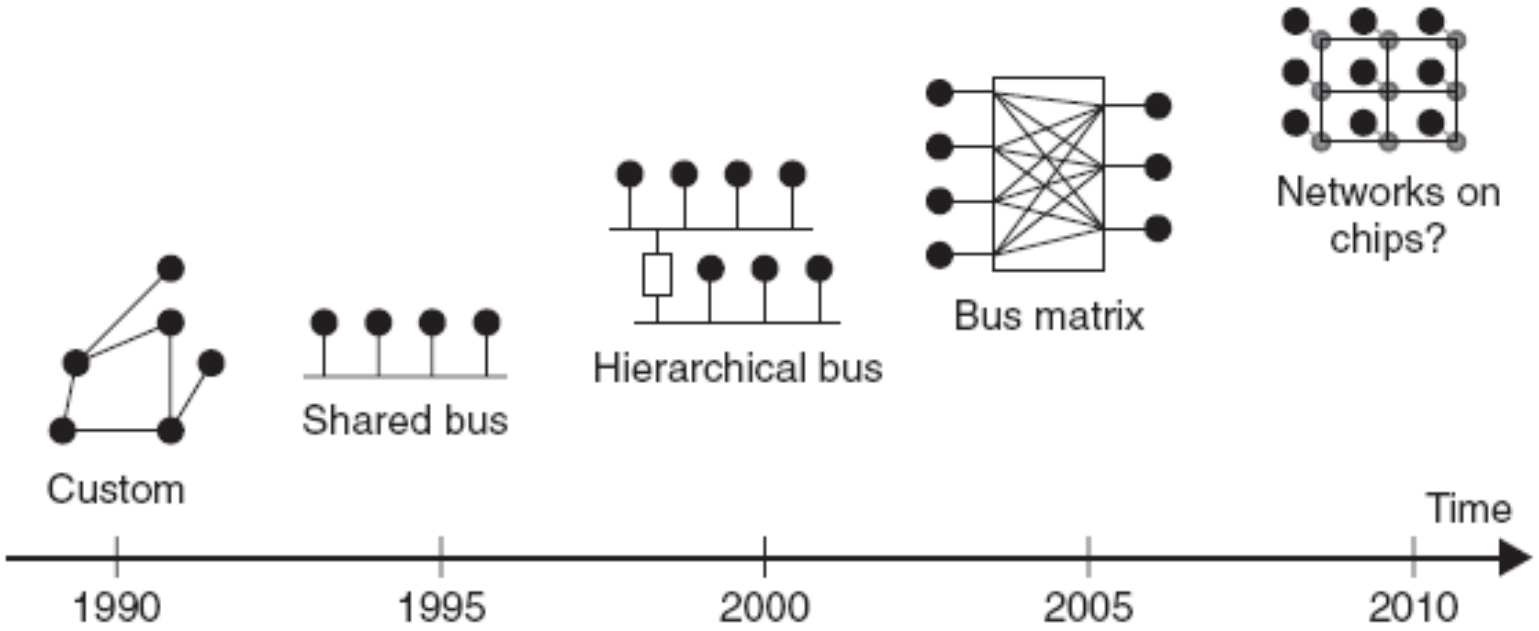
Basic Ingredients of a NoC

- N Computational Resources
  - Processing Elements (PE)
- 1 Connection Topology
- 1 Routing technique
- M ≤ N Switches
- N Network Interfaces
- 1 Addressing system
- 1 Communication Protocol
- 1 Programming model
  - Message passing
  - Shared Memory

Problems

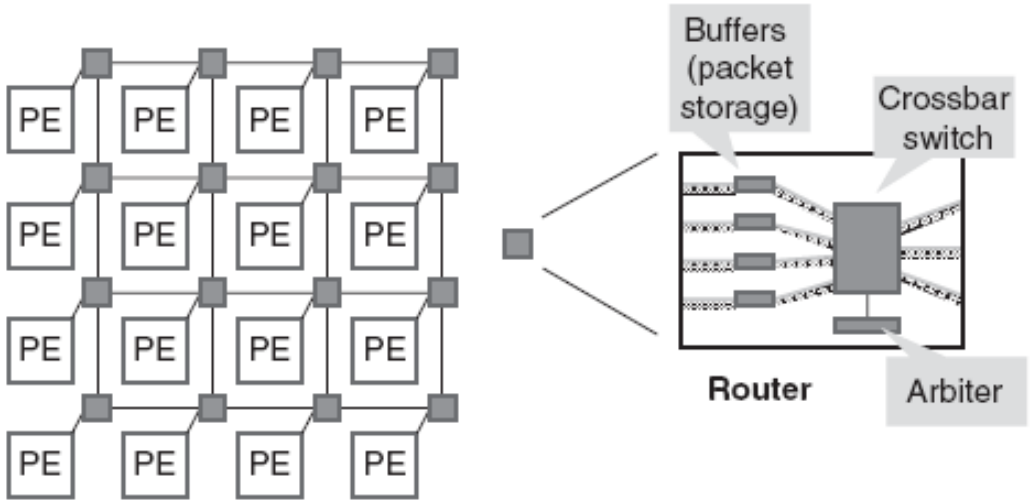
- ☹ Internal network contention causes (often unpredictable) latency.
- ☹ The network has a significant silicon area.
- ☹ Bus-oriented IPs need smart wrappers.
- ☹ Software needs clean synchronization in multiprocessor systems.
- ☹ System designers need reeducation for new concepts.

- Evolution of on-chip communication architectures



Introduction

- Network-on-chip (NoC) is a packet switched on-chip communication network designed using a layered methodology
  - "routes packets, not wires"
- NoCs use packets to route data from the source to the destination PE via a network fabric that consists of
  - switches (routers)
  - interconnection links (wires)



Introduction

- NoCs are an attempt to scale down the concepts of largescale networks, and apply them to the embedded system-on-chip (SoC) domain
- NoC Properties
  - Regular geometry that is scalable
  - Flexible QoS guarantees
  - Higher bandwidth
  - Reusable components
    - Buffers, arbiters, routers, protocol stack
  - No long global wires (or global clock tree)
    - No problematic global synchronization
    - GALS: Globally asynchronous, locally synchronous design
  - Reliable and predictable electrical and physical properties

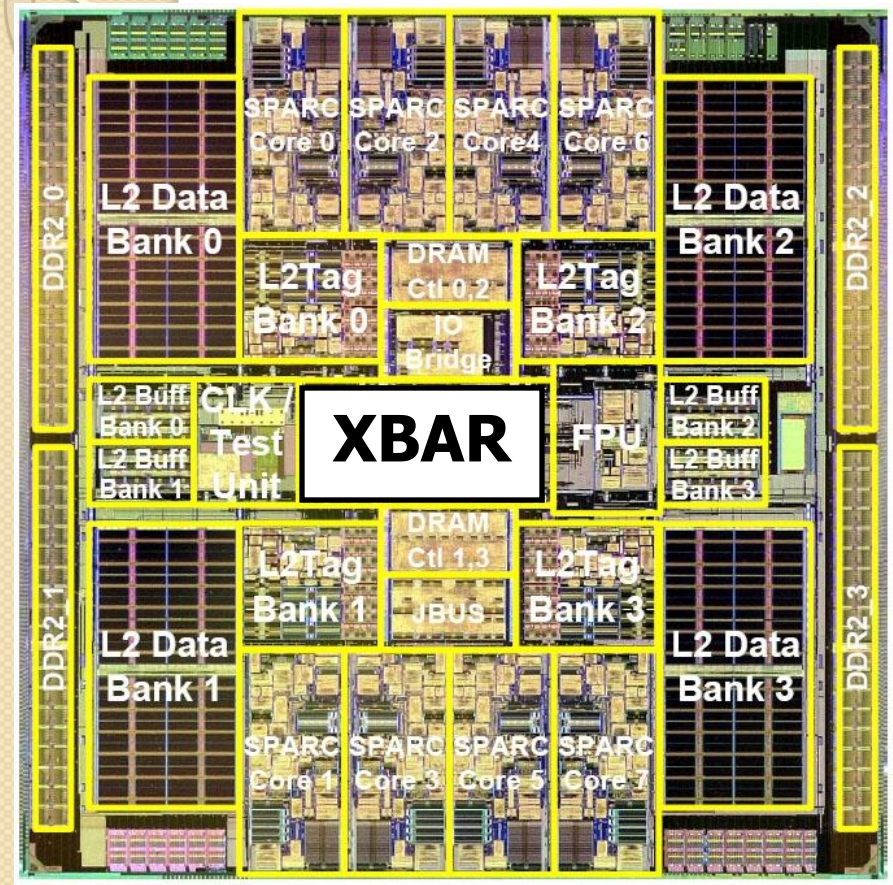
On-Chip Networks (NoCs)

- Why Network on Chip?
  - Ad-hoc wiring does not scale beyond a small number of cores
    - Prohibitive area
    - Long latency
- NoC offers
  - scalability
  - efficient multiplexing of communication
  - often modular in nature (ease verification)

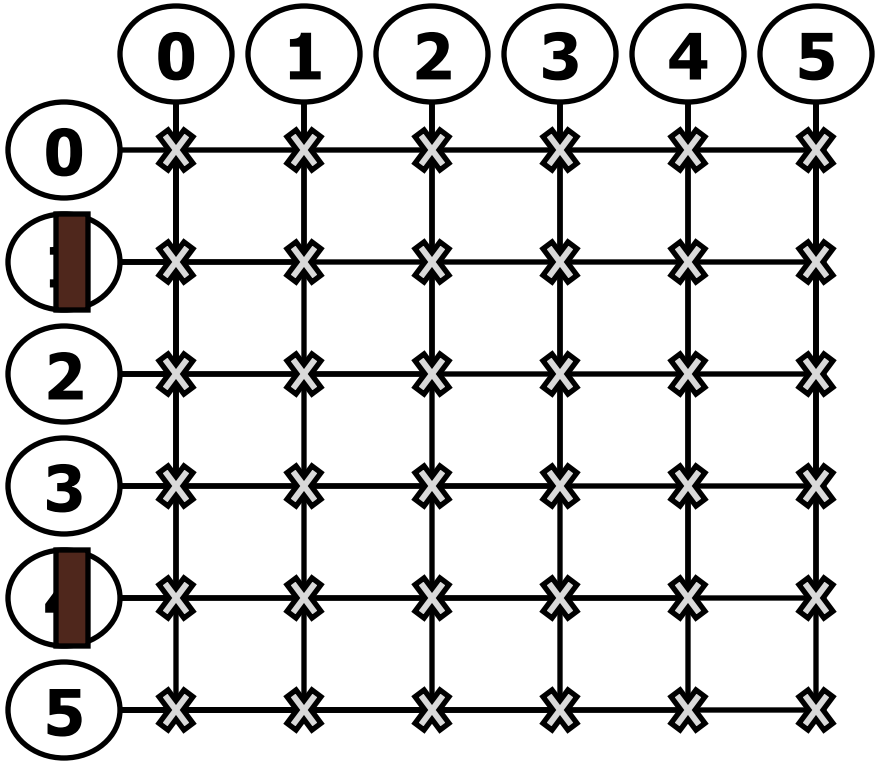
Differences between on-chip and off-chip networks

- Off-chip: I/O bottlenecks
  - Pin-limited bandwidth
  - Inherent overheads of off-chip I/O transmission
- On-chip
  - Tight area and power budgets
  - Ultra-low on-chip latencies

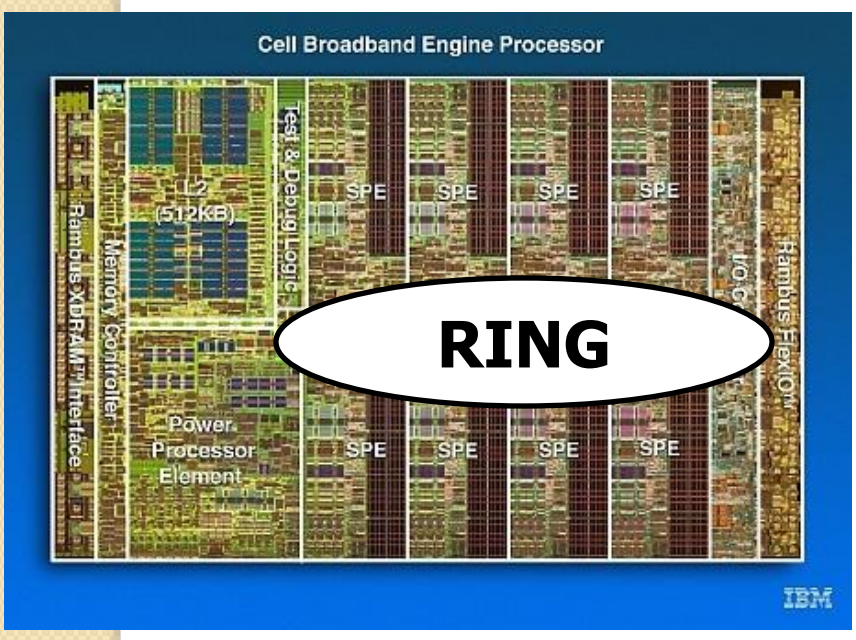
MulticoreExamples (1)



Sun Niagara



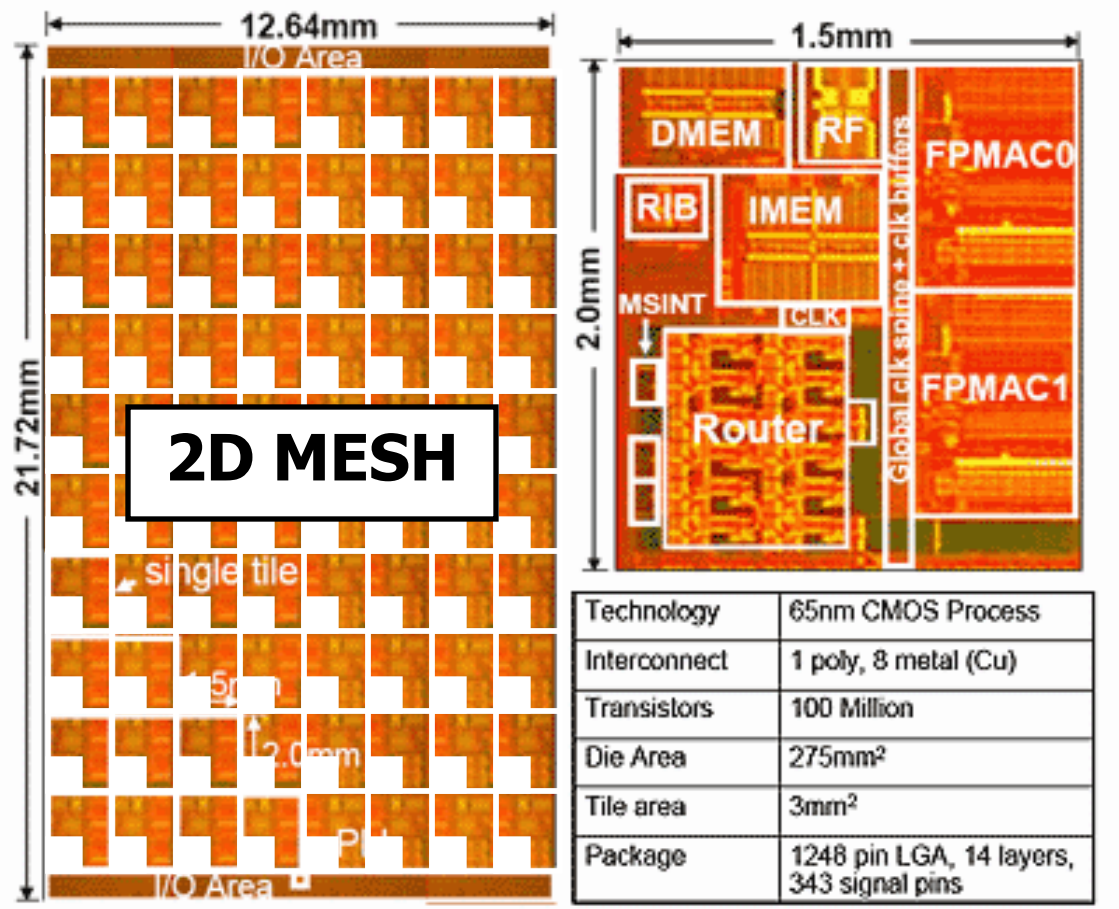
MulticoreExamples (2)



IBM Cell

- Element Interconnect Bus
  - 4 rings
  - Packet size: 16B-128B
  - Credit-based flow control
  - Up to 64 outstanding requests
  - Latency: 1 cycle/hop

Many Core Example



Technology	65nm CMOS Process
Interconnect	1 poly, 8 metal (Cu)
Transistors	100 Million
Die Area	275mm²
Tile area	3mm²
Package	1248 pin LGA, 14 layers, 343 signal pins

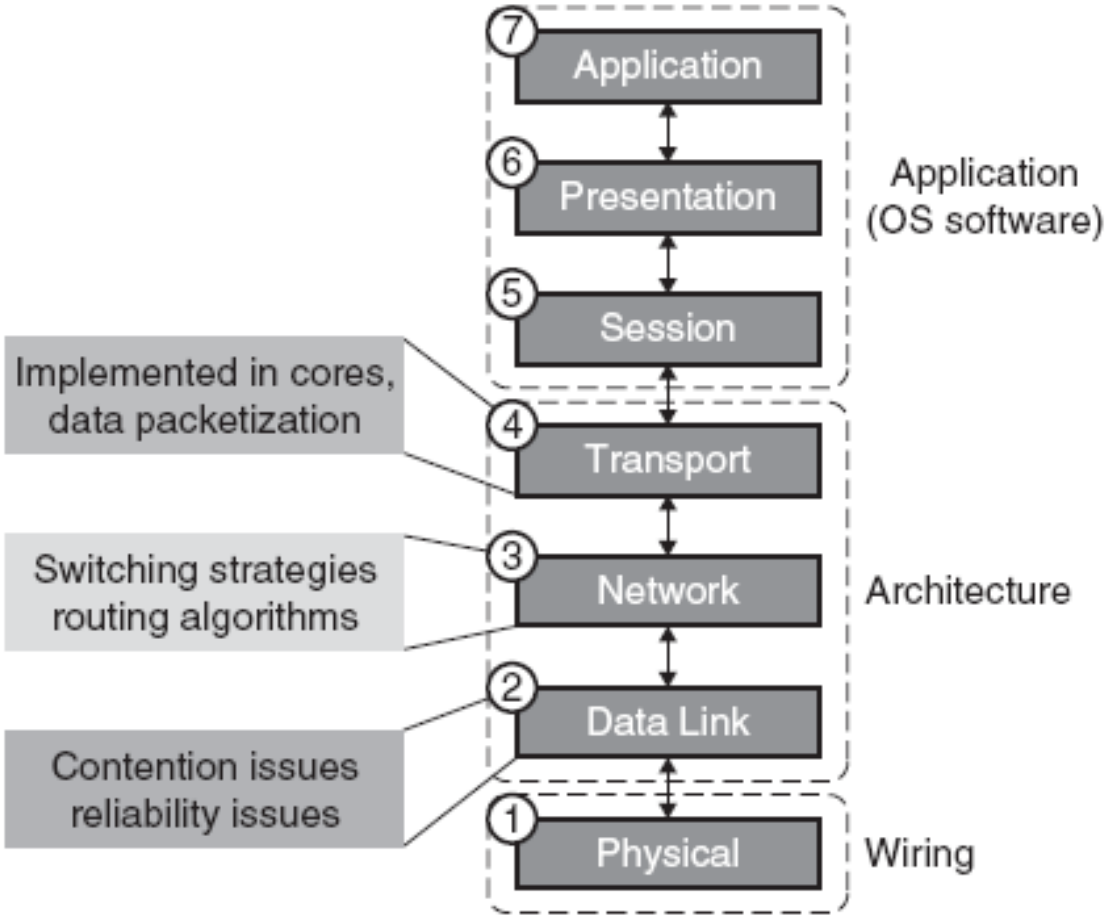
- Intel Polaris
  - 80 core prototype
- Academic Research ex:
  - MIT Raw, TRIPs
    - 2-D Mesh Topology
    - Scalar Operand Networks

References

- William Dally and Brian Towles. Principles and Practices of Interconnection Net- works. Morgan Kaufmann Pub., San Francisco, CA, 2003.
- William Dally and Brian Towles, "Route packets not wires: On-chip interconnection networks," in Proceedings of the 38th Annual Design Automation Conference (DAC-38), 2001, pp. 684–689.
- David Wentzlaff, Patrick Griffin, Henry Hoffman, LieweiBao, Bruce Edwards, Carl Ramey, Matthew Mattina, Chi-Chang Miao, John Brown III, and AnantAgarwal. On-chip interconnection architecture of the tile processor. IEEE Micro, pages 15– 31, 2007.
- Michael Bedford Taylor, Walter Lee, SamanAmarasinghe, and AnantAgarwal. Scalar operand networks: On-chip interconnect for ILP in partitioned architectures. In Proceedings of the International Symposium on High Performance Computer Architecture, February 2003.
- S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar. An 80-tile 1.28tflops network-on-chip in 65nm cmos. *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International, pages 98–589, Feb. 2007.*

Introduction

- ISO/OSI network protocol stack model



Outline

- Introduction
- NoC Topology
- Switching strategies
- Routing algorithms
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- Status and Open Problems

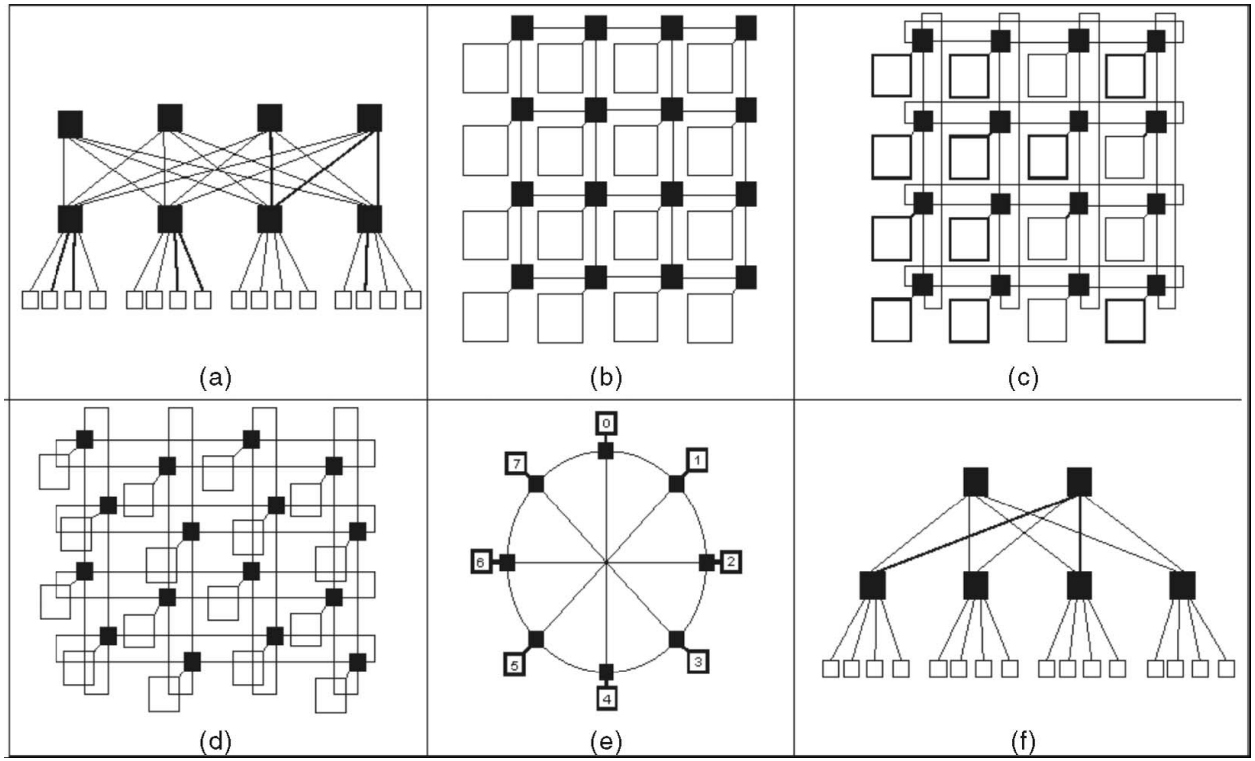
Topology Overview

- Definition: determines arrangement of channels and nodes in network
- Analogous to road map
- Often first step in network design
- Routing and flow control build on properties of topology

Topologies

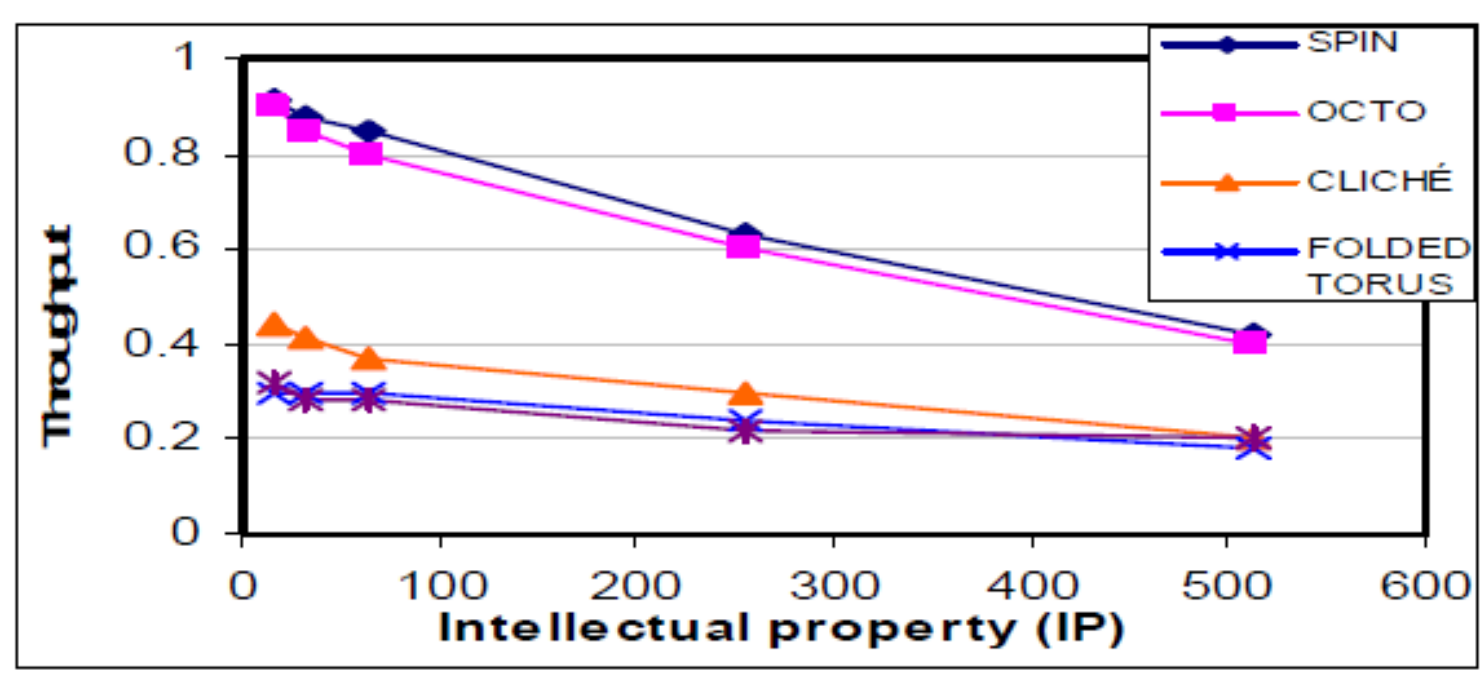
- Heritage of networks with new constraints
  - Need to accommodate interconnects in a 2D layout
  - Cannot route long wires (clock frequency bound)

a) SPIN,
b) CLICHE'
c) Torus
d) Folded torus
e) Octagon
f) BFT.



Topologies

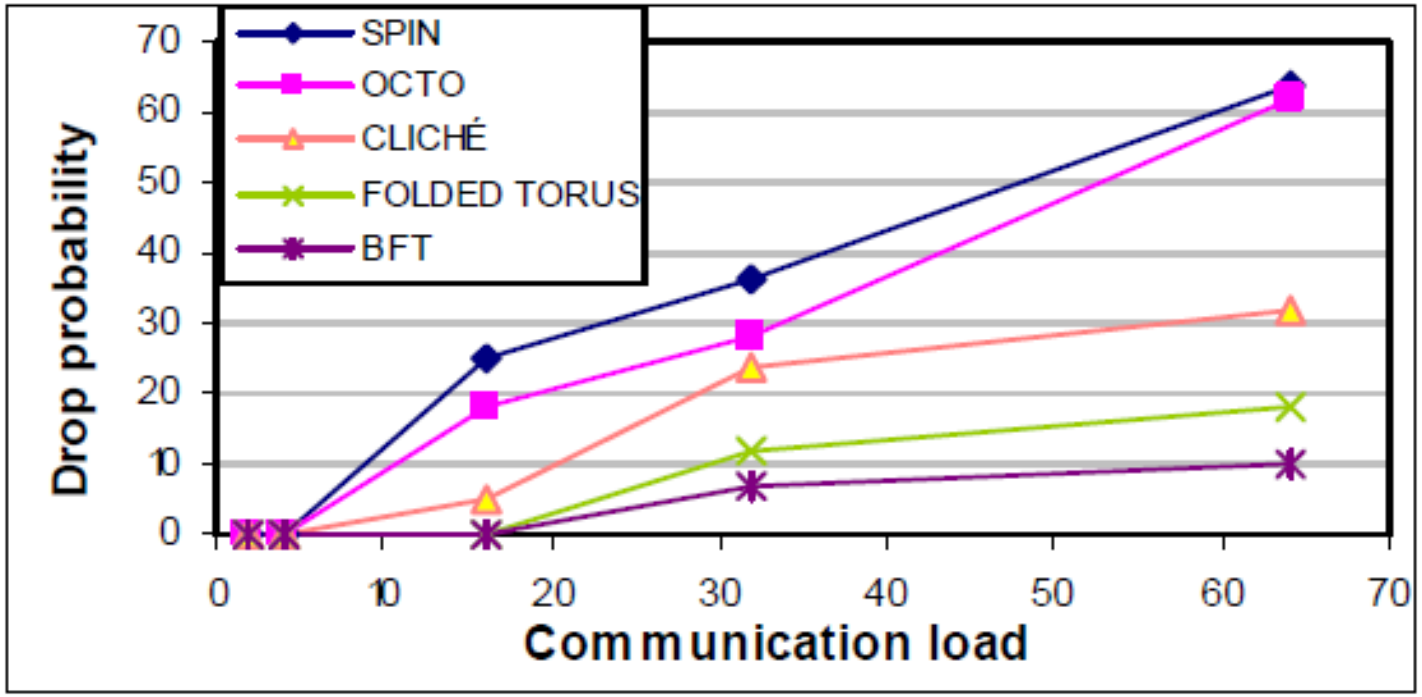
- Comparison of topologies according to different QoS parameters.



Throughput as a function of number of IPs.

Topologies

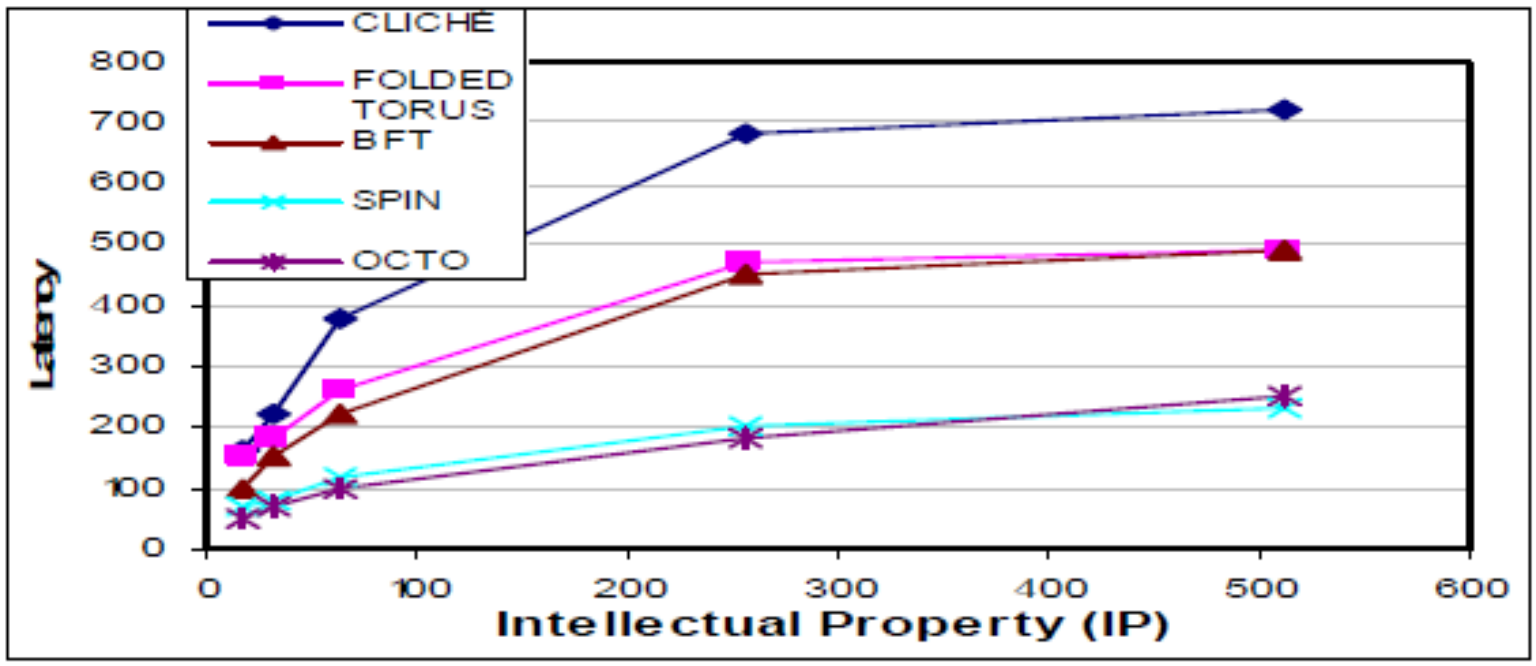
- Comparison of topologies according to different QoS parameters.



Drop probability as a function of number of IPs.

Topologies

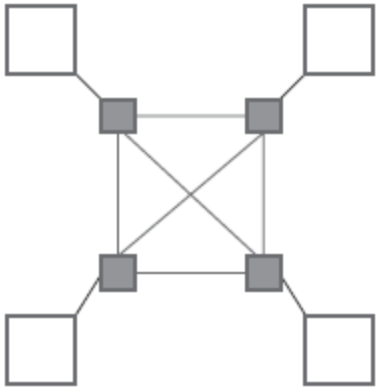
- Comparison of topologies according to different QoS parameters.



Latency as a function of number of IPs.

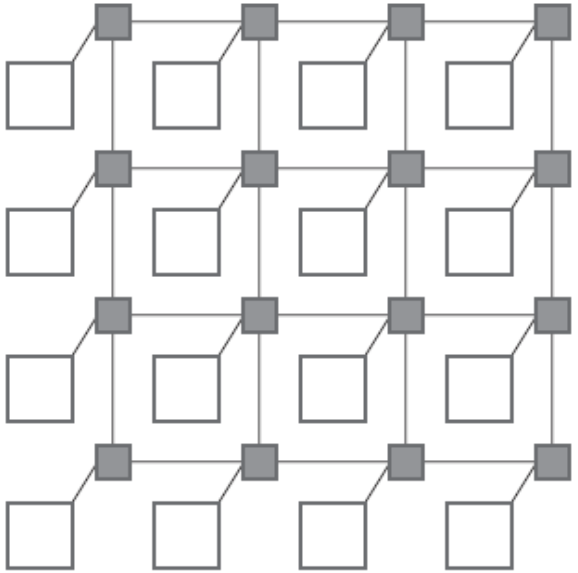
NoC Topology

- Direct Topologies
  - each node has direct point-to-point link to a subset of other nodes in the system called neighboring nodes
  - nodes consist of computational blocks and/or memories, as well as a NI block that acts as a router
  - e.g. Nostrum, SOCBUS, Proteo, Octagon
  - as the number of nodes in the system increases, the total available communication bandwidth also increases
  - fundamental trade-off is between connectivity and cost



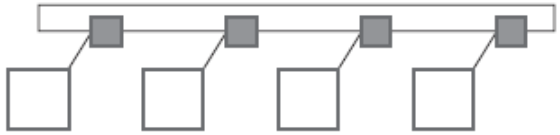
NoC Topology

- Most direct network topologies have an orthogonal implementation, where nodes can be arranged in an n-dimensional orthogonal space
  - routing for such networks is fairly simple
  - e.g. n-dimensional mesh, torus, folded torus, hypercube, and octagon
- 2D mesh is most popular topology
  - all links have the same length
    - eases physical design
  - area grows linearly with the number of nodes
  - must be designed in such a way as to avoid traffic accumulating in the center of the mesh

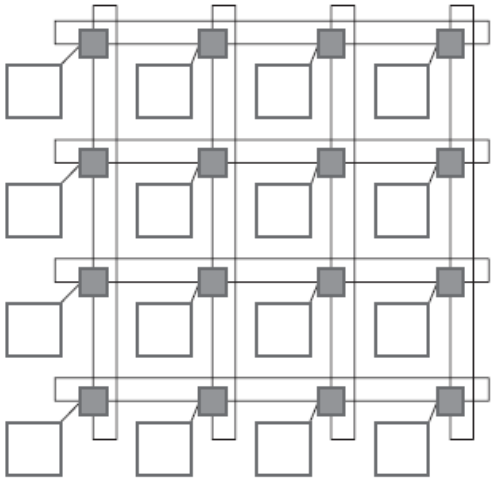


NoC Topology

- Torus topology, also called a k-ary n-cube, is an n-dimensional grid with k nodes in each dimension
  - k-ary 1-cube (1-D torus) is essentially a ring network with k nodes
    - limited scalability as performance decreases when more nodes

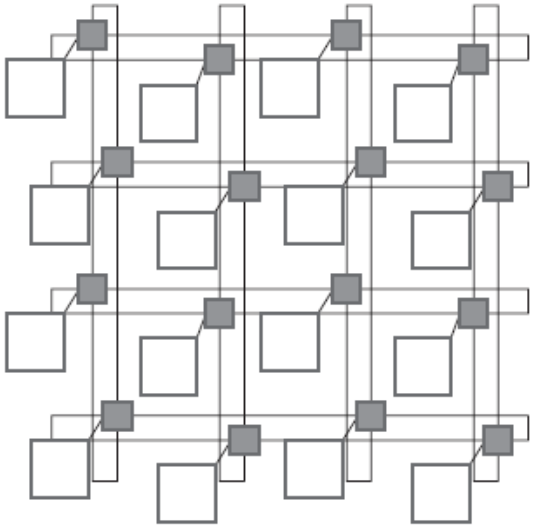


  - k-ary 2-cube (i.e., 2-D torus) topology is similar to a regular mesh
    - except that nodes at the edges are connected to switches at the opposite edge via wrap-around channels
    - long end-around connections can, however, lead to excessive delays



NoC Topology

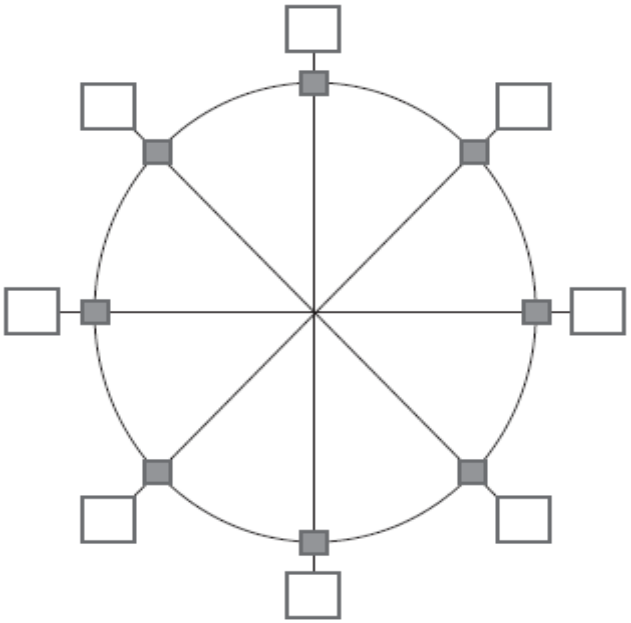
- Folding torus topology overcomes the long link limitation of a 2-D torus
  - links have the same size



- Meshes and tori can be extended by adding bypass links to increase performance at the cost of higher area

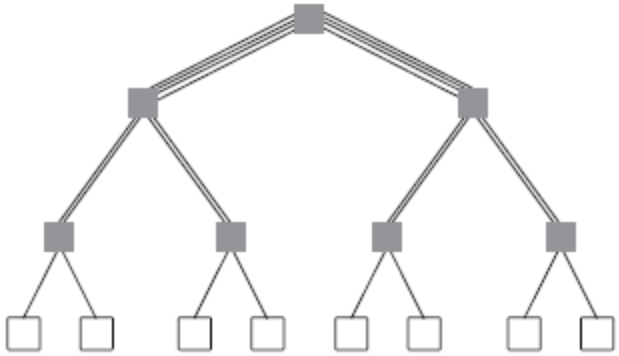
NoC Topology

- Octagon topology is another example of a direct network
  - messages being sent between any 2 nodes require at most two hops
  - more octagons can be tiled together to accommodate larger designs
    - by using one of the nodes is used as a bridge node



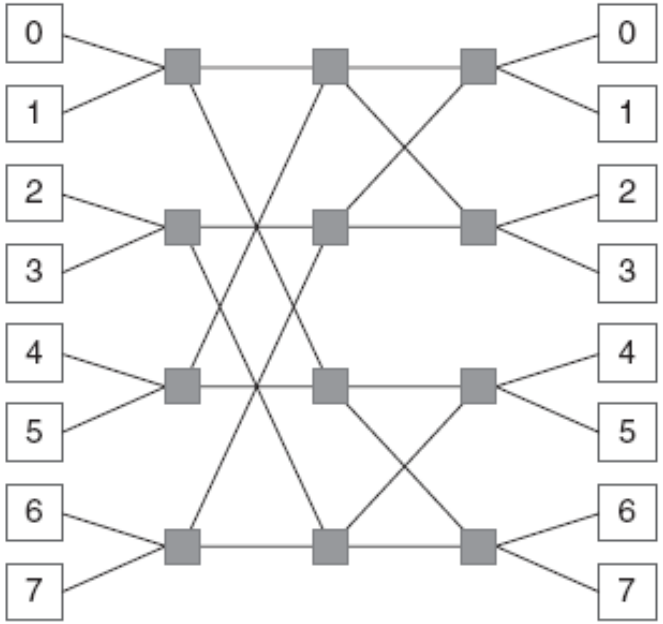
NoC Topology

- Indirect Topologies
  - each node is connected to an external switch, and switches have point-to-point links to other switches
  - switches do not perform any information processing, and correspondingly nodes do not perform any packet switching
  - e.g. SPIN, crossbar topologies
- Fat tree topology
  - nodes are connected only to the leaves of the tree
  - more links near root, where bandwidth requirements are higher



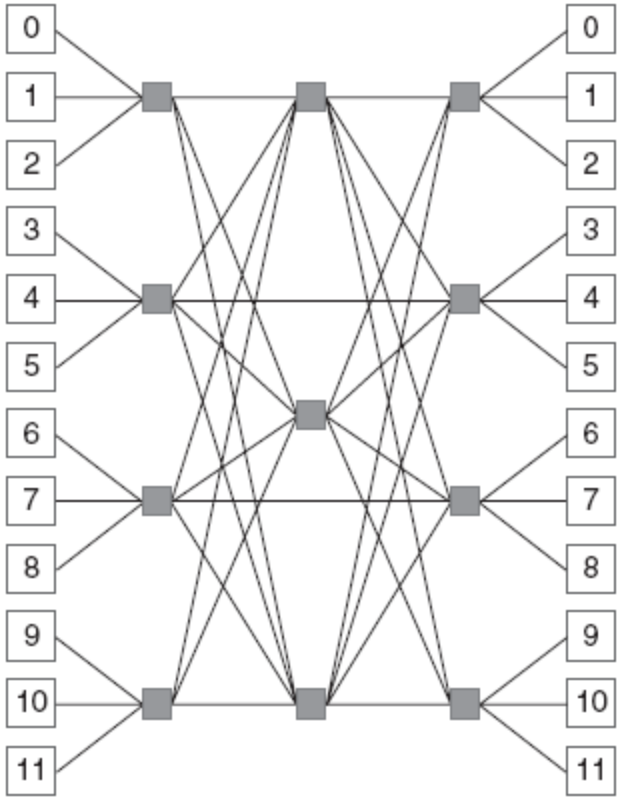
NoC Topology

- k-ary n-fly butterfly network
  - blocking multi-stage network – packets may be temporarily blocked or dropped in the network if contention occurs
  - k^n nodes, and n stages of k^{n-1} k x k crossbar
  - e.g. 2-ary 3-fly butterfly network



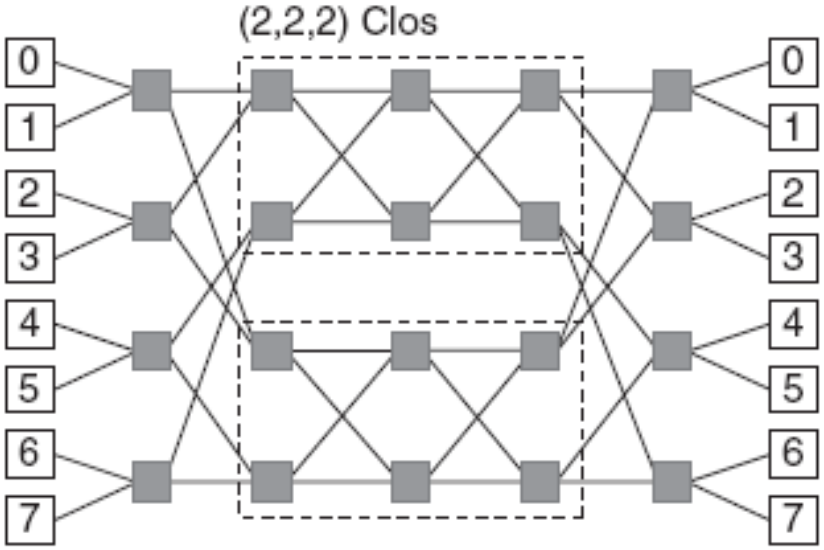
NoC Topology

- (m, n, r) symmetric Clos network
  - three-stage network in which each stage is made up of a number of crossbar switches
  - m is the no. of middle-stage switches
  - n is the number of input/output nodes on each input/output switch
  - r is the number of input and output switches
  - e.g. (3, 3, 4) Clos network
  - non-blocking network
  - expensive (several full crossbars)



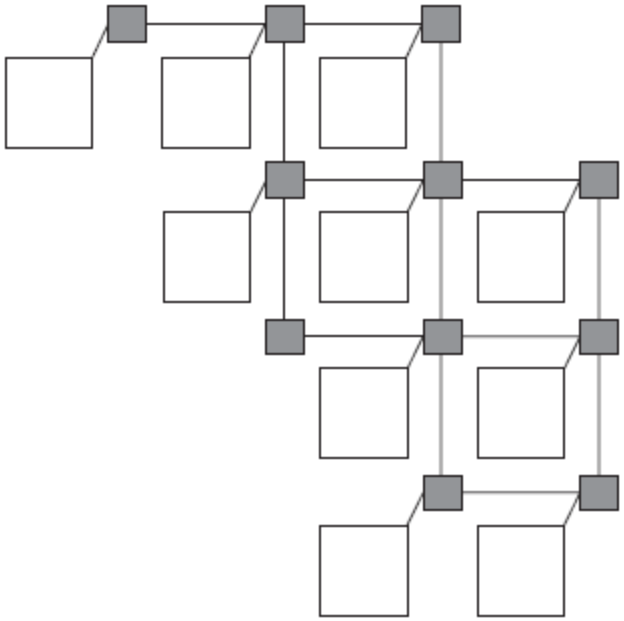
NoC Topology

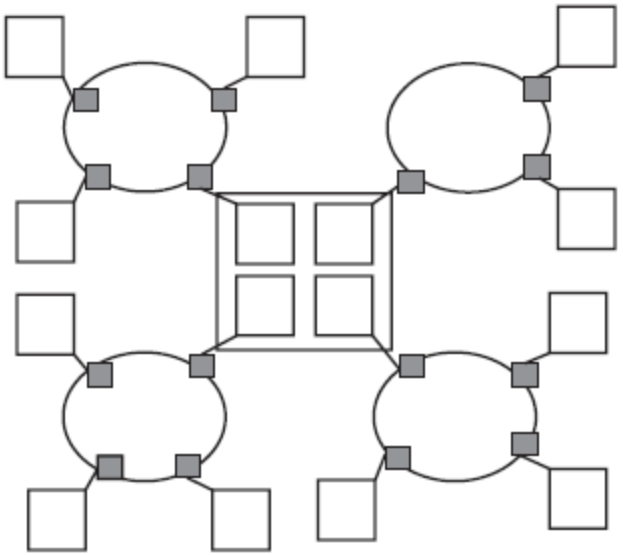
- Benes network
  - rearrangeable network in which paths may have to be rearranged to provide a connection, requiring an appropriate controller
  - Clos topology composed of 2 x 2 switches
  - e.g. (2, 2, 4) re-arrangeable Clos network constructed using two (2, 2, 2) Clos networks with 4 x 4 middle switches



NoC Topology

- Irregular or ad hoc network topologies
  - customized for an application
  - usually a mix of shared bus, direct, and indirect network topologies
  - e.g. reduced mesh, cluster-based hybrid topology





Path Diversity

- Multiple minimum length paths between source and destination pair
- Fault tolerance
- Better load balancing in network
- Routing algorithm should be able to exploit path diversity
- We'll see shortly
  - Butterfly has no path diversity
  - Torus can exploit path diversity

Path Diversity (2)

- Edge disjoint paths: no links in common
- Node disjoint paths: no nodes in common except source and destination
- If j = minimum number of edge/node disjoint paths between any source-destination pair
  - Network can tolerate j link/node failures

- Path diversity does not provide pt-to-pt order
  - Implications on coherence protocol design!

Symmetry

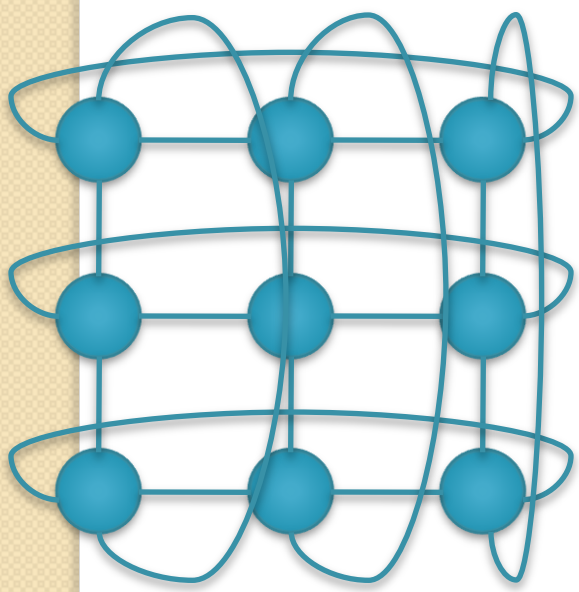
- Vertex symmetric:
  - An automorphism exists that maps any node a onto another node b
  - Topology same from point of view of all nodes
- Edge symmetric:
  - An automorphism exists that maps any channel a onto another channel b

Direct & Indirect Networks

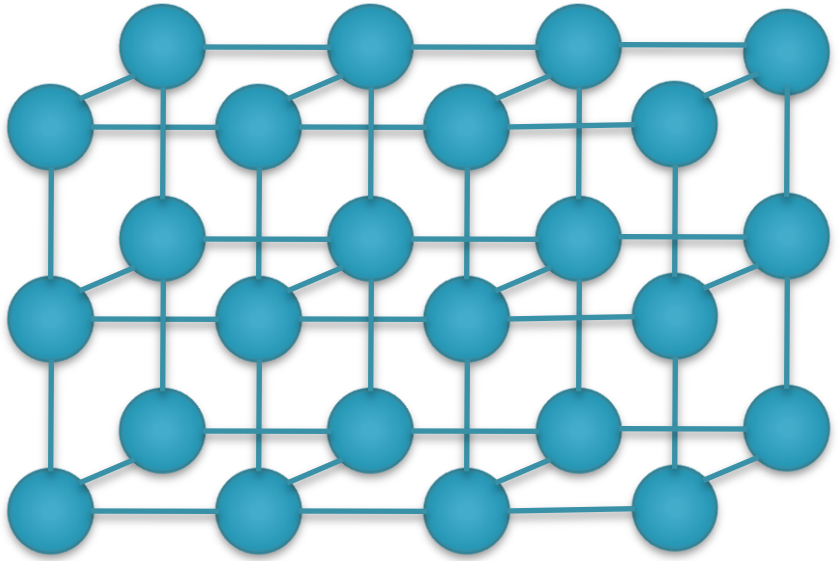
- Direct: Every switch also network end point
  - Ex: Torus
- Indirect: Not all switches are end points
  - Ex: Butterfly

Torus (1)

- K-ary n-cube: k^n network nodes
- n-dimensional grid with k nodes in each dimension

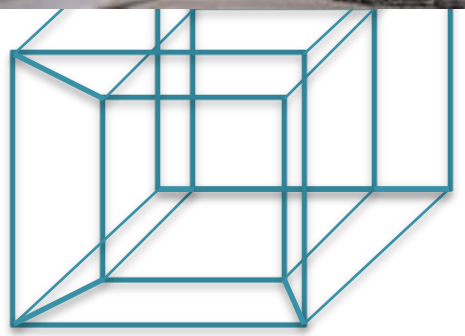


3-ary 2-mesh



2,3,4-ary 3-mesh

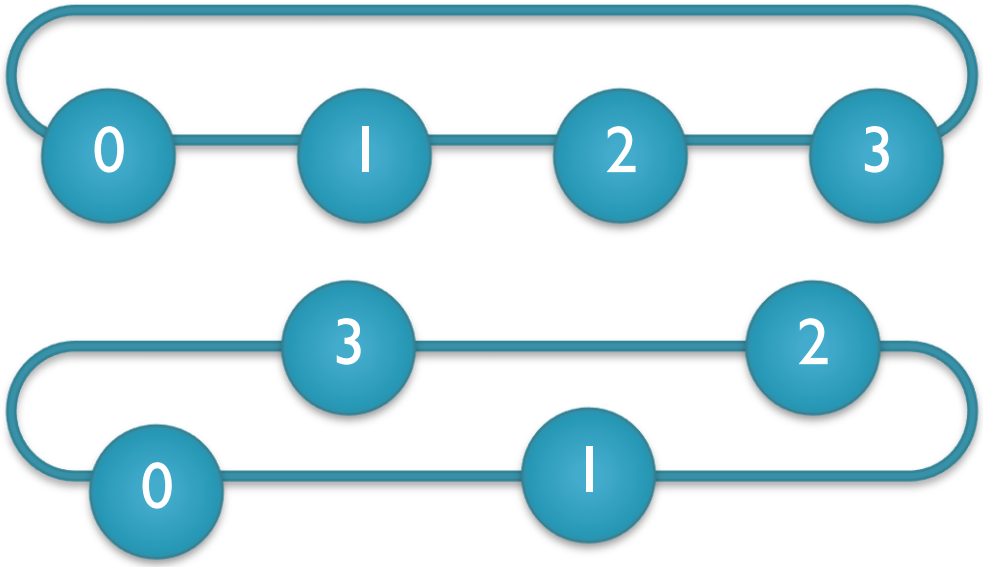
Torus (2)



- Topologies in Torus Family
  - Ring k-ary 1-cube
  - Hypercubes 2-ary n-cube
- Edge Symmetric
  - Good for load balancing
  - Removing wrap-around links for mesh loses edge symmetry
    - More traffic concentrated on center channels
- Good path diversity
- Exploit locality for near-neighbor traffic

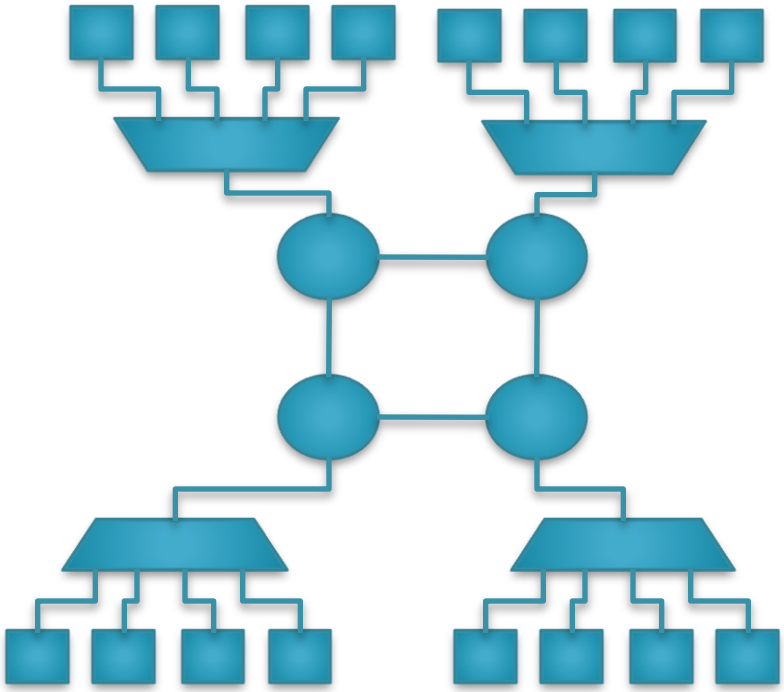
Implementation

- Folding
  - Equalize path lengths
    - Reduces max link length
    - Increases length of other links



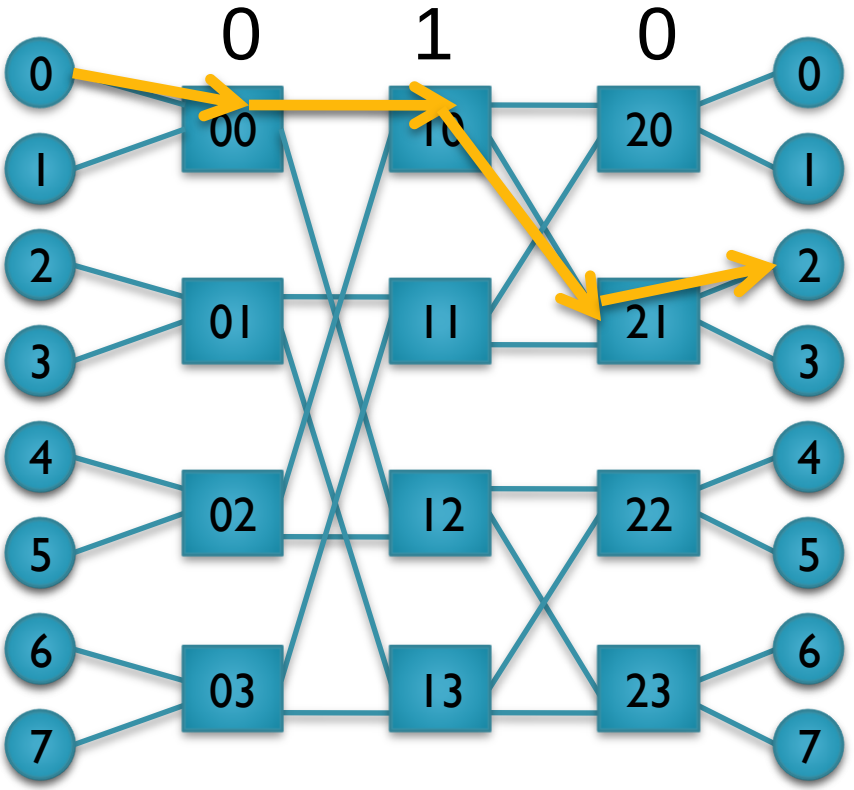
Concentration

- Don't need 1:1 ratio of network nodes and cores/memory
- Ex: 4 cores concentrated to 1 router



Butterfly

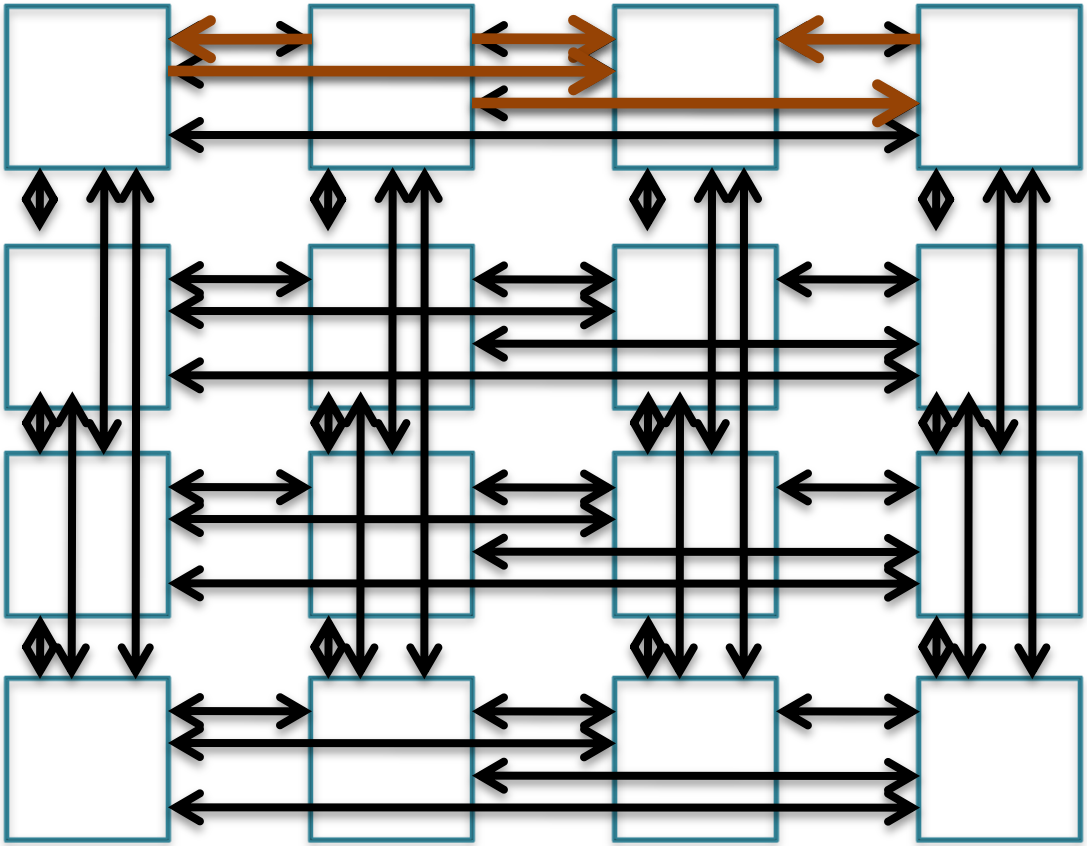
- K-ary n-fly: k^n network nodes
- Example: 2-ary 3-fly
- Routing from 000 to 010
  - Dest address used to directly route packet
  - Bit n used to select output port at stage n



Flattened Butterfly

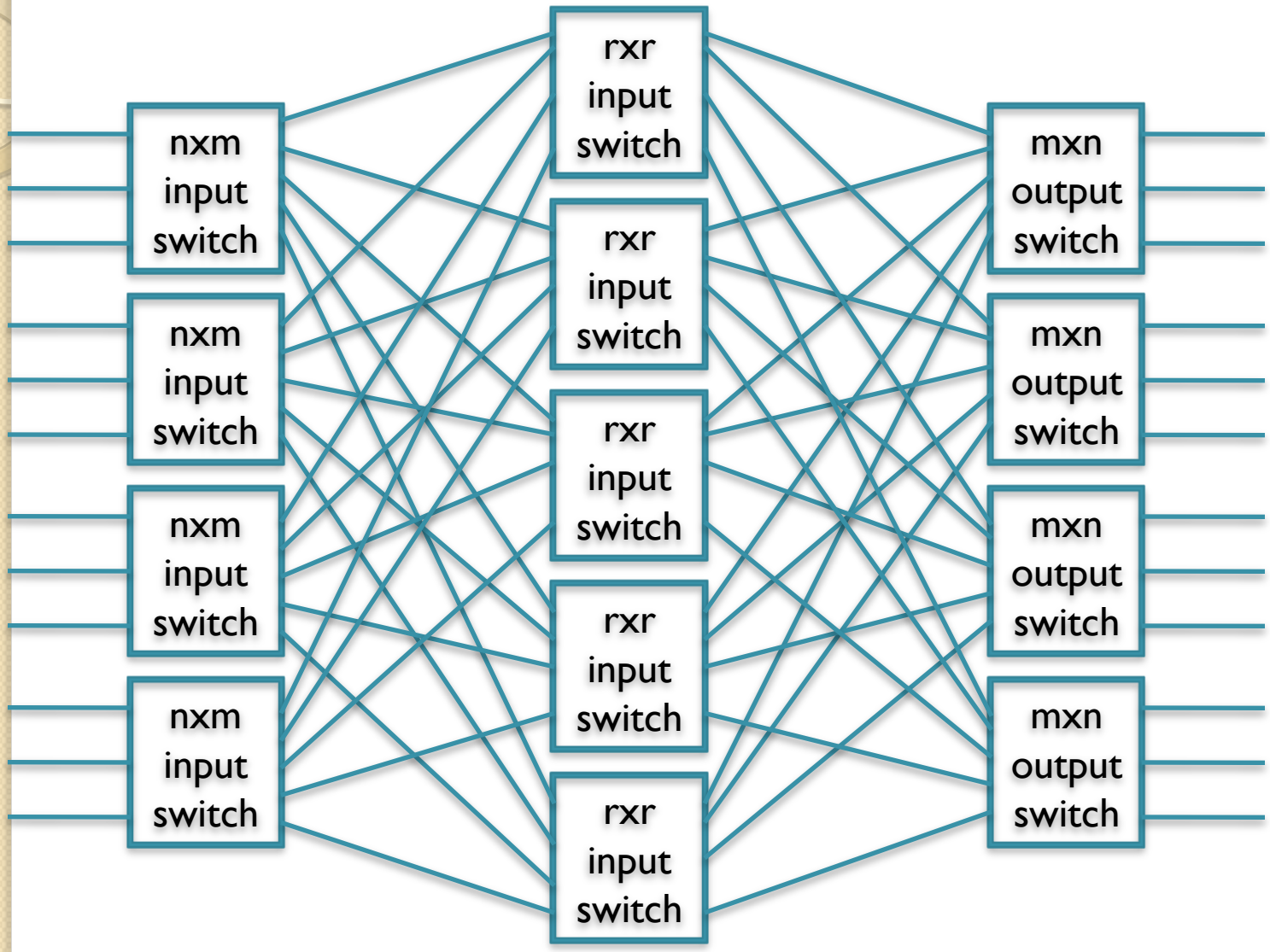
- Proposed by Kim et al (ISCA 2007)
  - Adapted for on-chip (MICRO 2007)
- Advantages
  - Max distance between nodes = 2 hops
  - Lower latency and improved throughput compared to mesh
- Disadvantages
  - Requires higher port count on switches (than mesh, torus)
  - Long global wires
  - Need non-minimal routing to balance load

Flattened Butterfly



- Path diversity through non-minimal routes

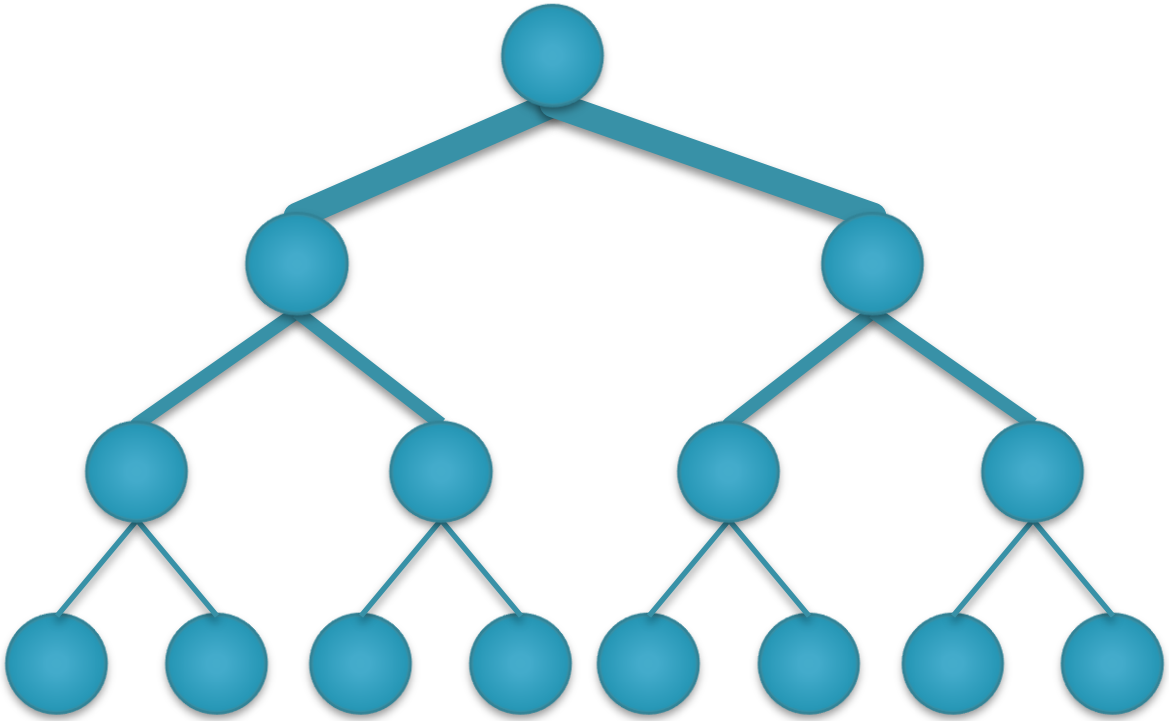
Clos Network



Clos Network

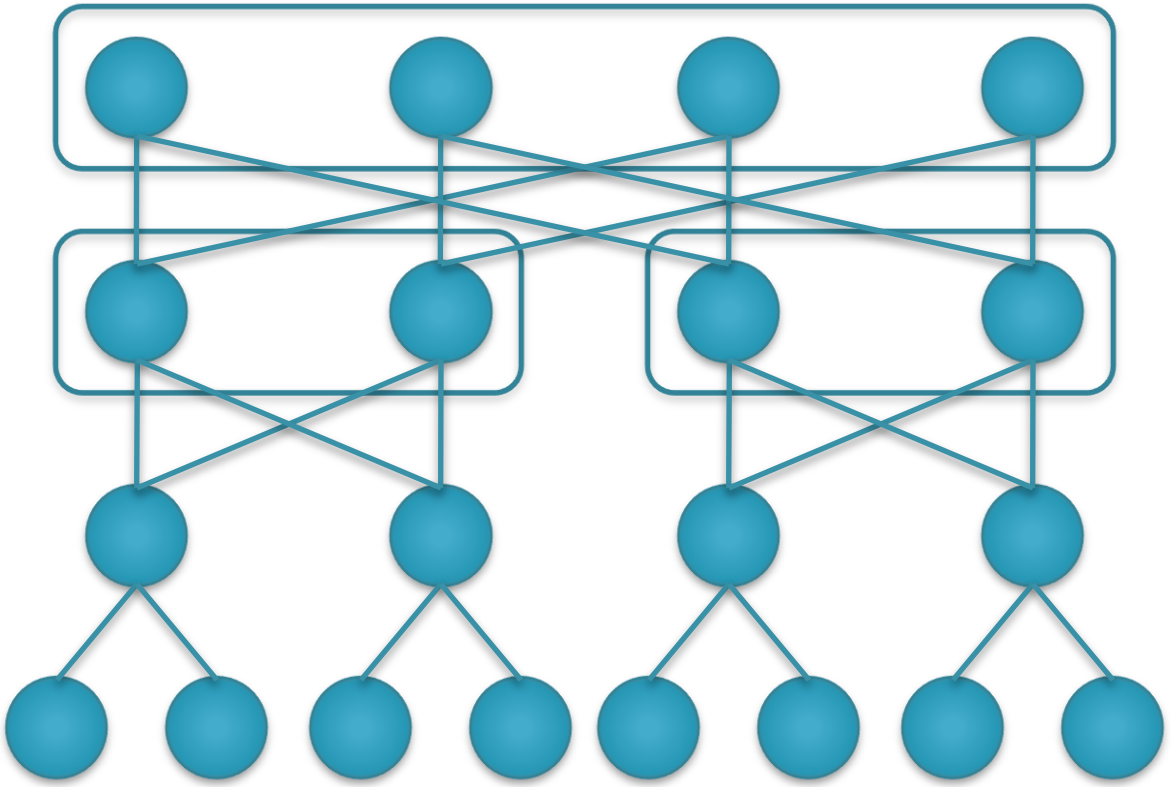
- 3-stage indirect network
- Characterized by triple (m, n, r)
  - M: # of middle stage switches
  - N: # of input/output ports on input/output switches
  - R: # of input/output switching
- Hop Count = 4

Folded Clos (Fat Tree)



- Bandwidth remains constant at each level
- Regular Tree: Bandwidth decreases closer to root

Fat Tree (2)



- Provides path diversity

Common On-Chip Topologies

- Torus family: mesh, concentrated mesh, ring
  - Extending to 3D stacked architectures
  - Favored for low port count switches
- Butterfly family: Flattened butterfly

Topology Summary

- First network design decision
- Critical impact on network latency and throughput
  - Hop count provides first order approximation of message latency
  - Bottleneck channels determine saturation throughput

Outline

- Introduction
- NoC Topology
- Switching strategies
- Routing algorithms
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- Status and Open Problems

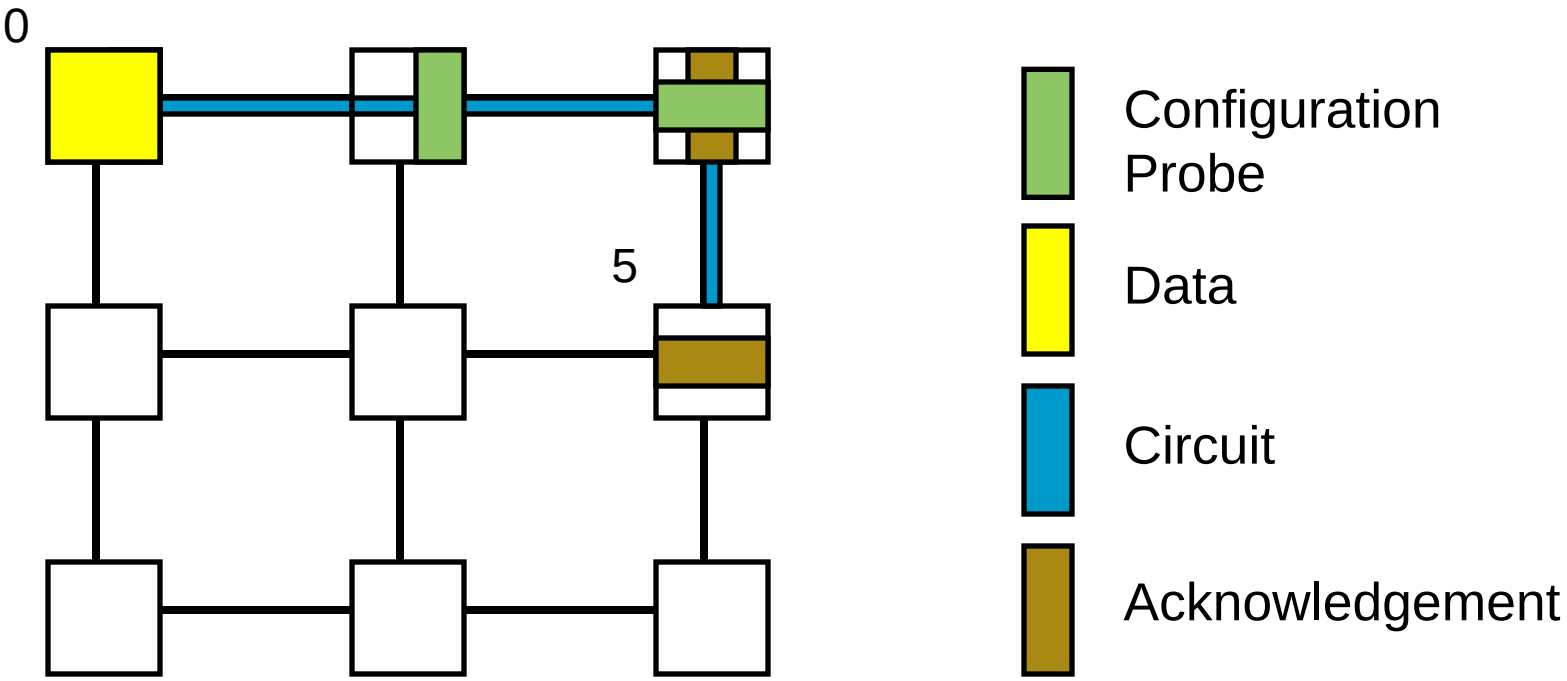
Switching

- Different flow control techniques based on granularity
- Circuit-switching: operates at the granularity of messages
- Packet-based: allocation made to whole packets
- Flit-based: allocation made on a flit-by-flit basis

Circuit Switching

- All resources (from source to destination) are allocated to the message prior to transport
  - Probe sent into network to reserve resources
- Once probe sets up circuit
  - Message does not need to perform any routing or allocation at each network hop
  - Good for transferring large amounts of data
    - Can amortize circuit setup cost by sending data with very low per-hop overheads
- No other message can use those resources until transfer is complete
  - Throughput can suffer due setup and hold time for circuits

Circuit Switching Example

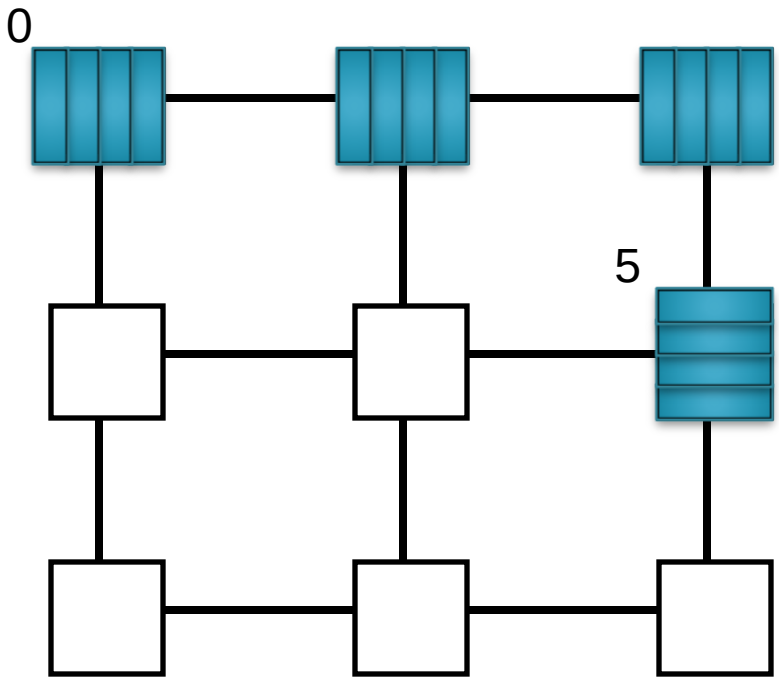


- Significant latency overhead prior to data transfer
- Other requests forced to wait for resources

Packet-based Flow Control

- Store and forward
- Links and buffers are allocated to entire packet
- Head flit waits at router until entire packet is buffered before being forwarded to the next hop
- Not suitable for on-chip
  - Requires buffering at each router to hold entire packet
  - Incurs high latencies (pays serialization latency at each hop)

Store and Forward Example

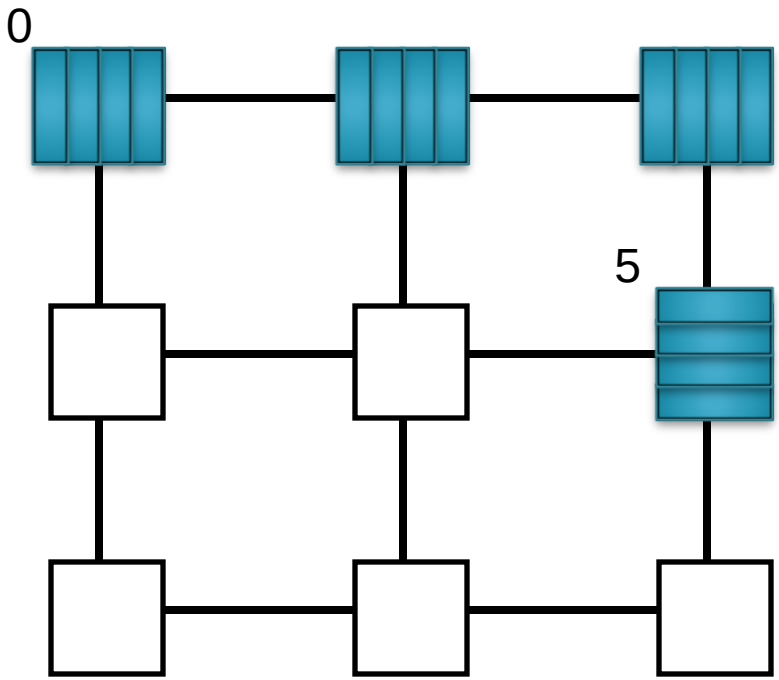


- High per-hop latency
- Larger buffering required

Virtual Cut Through

- Packet-based: similar to Store and Forward
- Links and Buffers allocated to entire packets
- Flits can proceed to next hop before tail flit has been received by current router
  - But only if next router has enough buffer space for entire packet
- Reduces the latency significantly compared to SAF
- But still requires large buffers
  - Unsuitable for on-chip

Virtual Cut Through Example

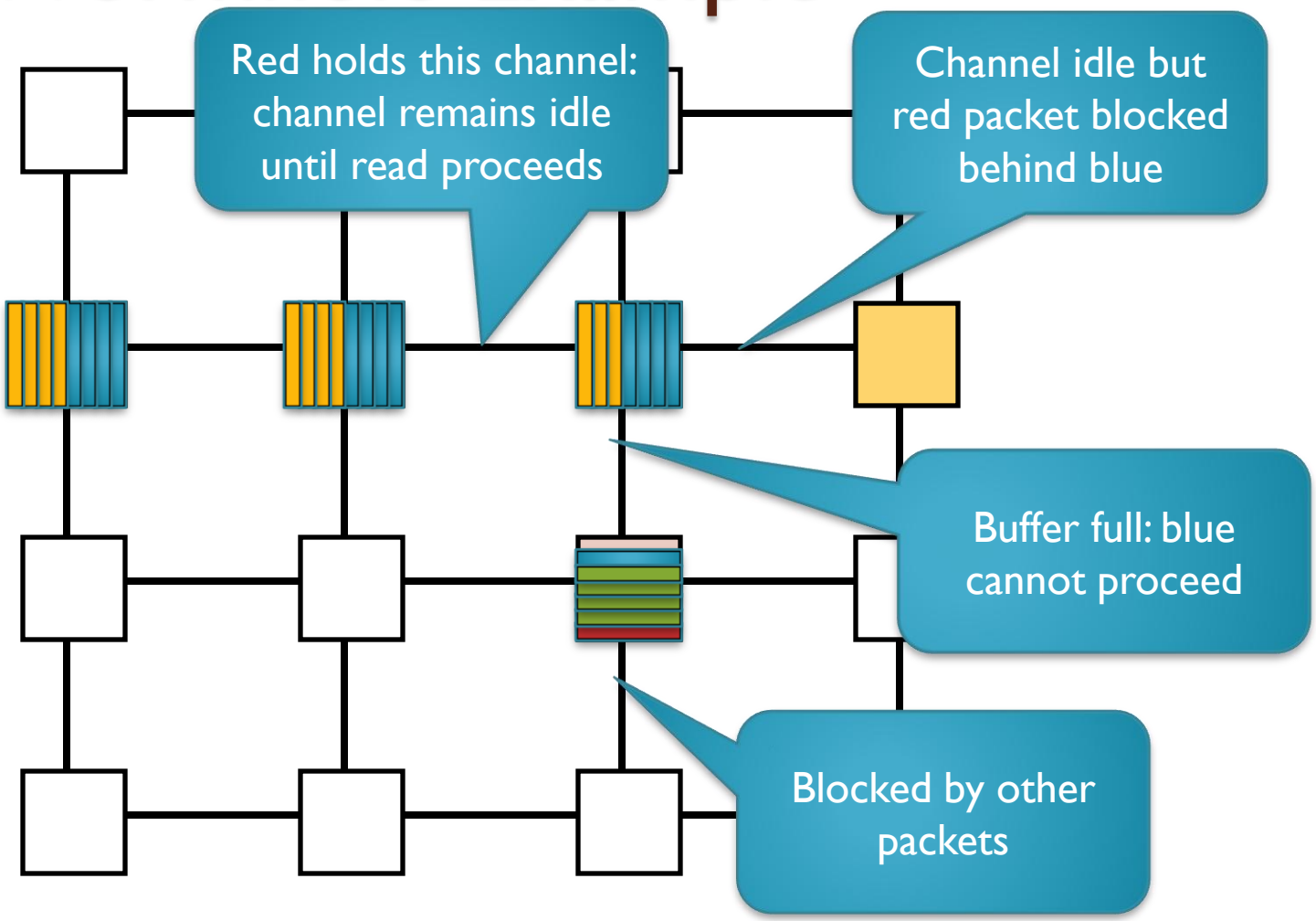


- Lower per-hop latency
- Larger buffering required

Flit Level Flow Control

- Wormhole flow control
- Flit can proceed to next router when there is buffer space available for that flit
  - Improved over SAF and VCT by allocating buffers on a flit-basis
- Pros
  - More efficient buffer utilization (good for on-chip)
  - Low latency
- Cons
  - Poor link utilization: if head flit becomes blocked, all links spanning length of packet are idle
    - Cannot be re-allocated to different packet
    - Suffers from head of line (HOL) blocking

Wormhole Example

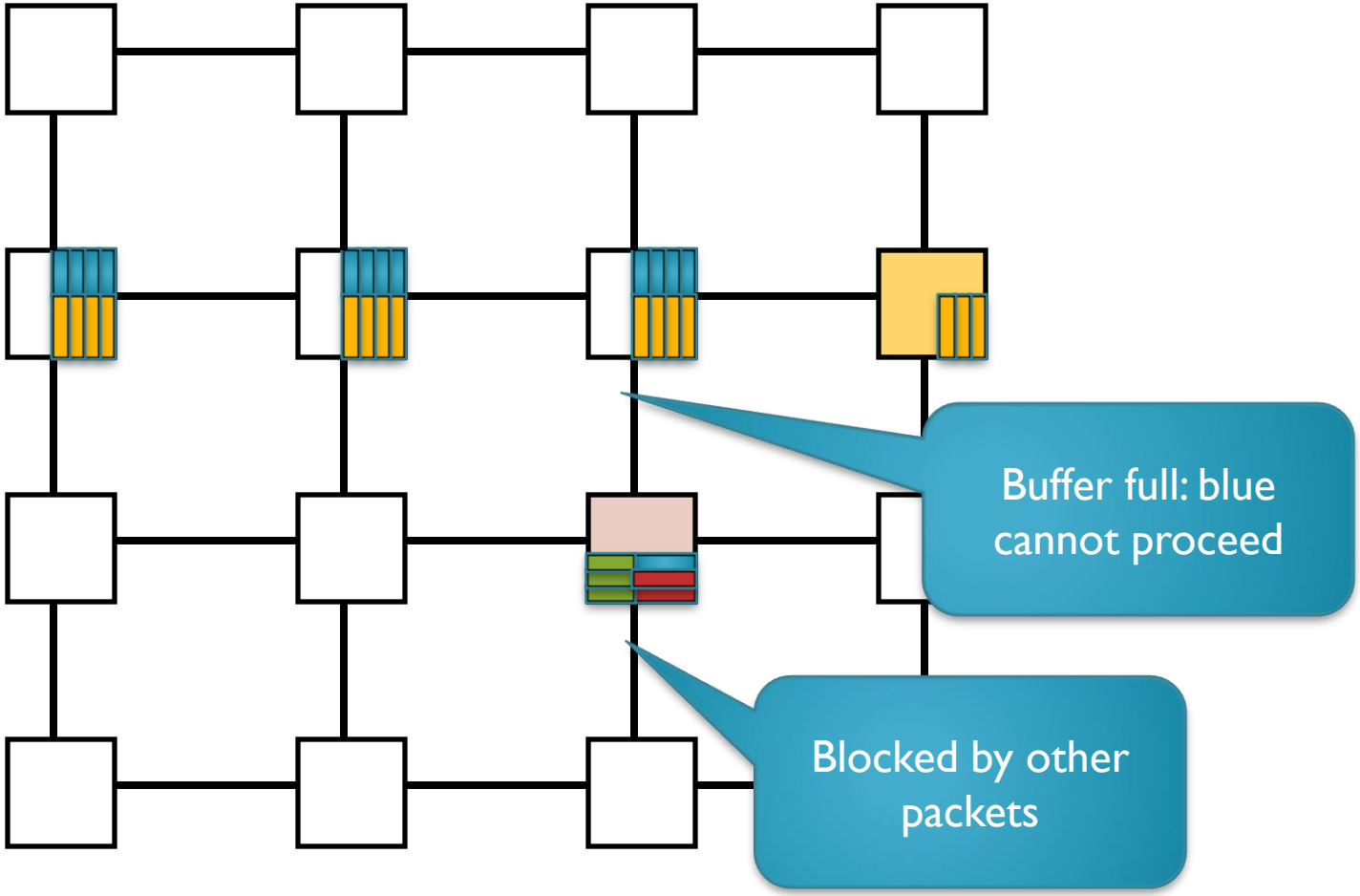


- 6 flit buffers/input port

Virtual Channel Flow Control

- Virtual channels used to combat HOL block in wormhole
- Virtual channels: multiple flit queues per input port
  - Share same physical link (channel)
- Link utilization improved
  - Flits on different VC can pass blocked packet

Virtual Channel Example



- 6 flit buffers/input port
- 3 flit buffers/VC

Deadlock

- Using flow control to guarantee deadlock freedom give more flexible routing
- Escape Virtual Channels
  - If routing algorithm is not deadlock free
  - VCs can break resource cycle
  - Place restriction on VC allocation or require one VC to be DOR
- Assign different message classes to different VCs to prevent protocol level deadlock
  - Prevent req-ack message cycles

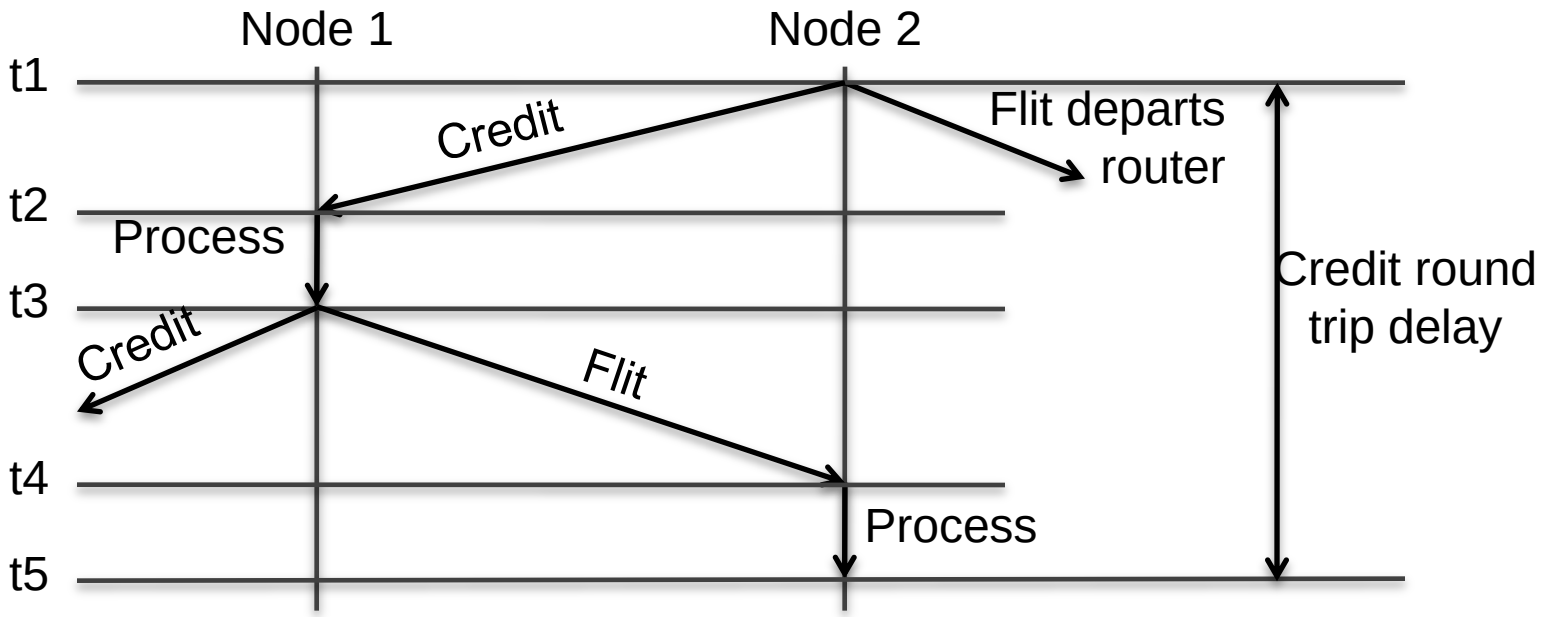
Buffer Backpressure

- Need mechanism to prevent buffer overflow
  - Avoid dropping packets
  - Upstream nodes need to know buffer availability at downstream routers
- Significant impact on throughput achieved by flow control
- Credits
- On-off

Credit-Based Flow Control

- Upstream router stores credit counts for each downstream VC
- Upstream router
  - When flit forwarded
    - Decrement credit count
  - Count == 0, buffer full, stop sending
- Downstream router
  - When flit forwarded and buffer freed
    - Send credit to upstream router
    - Upstream increments credit count

Credit Timeline

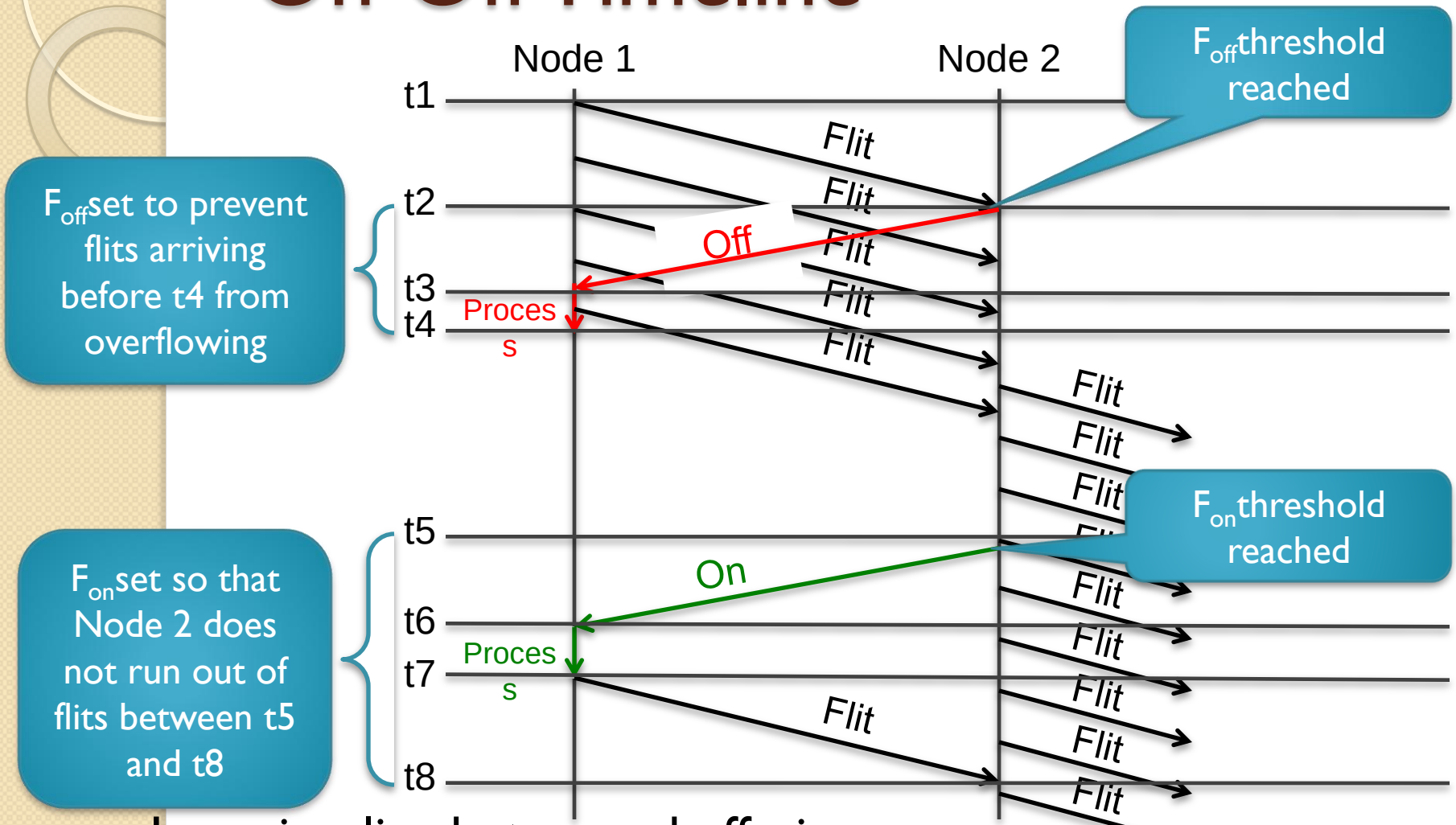


- Round-trip credit delay:
  - Time between when buffer empties and when next flit can be processed from that buffer entry
  - If only single entry buffer, would result in significant throughput degradation
  - Important to size buffers to tolerate credit turn-around

On-Off Flow Control

- Credit: requires upstream signaling for every flit
- On-off: decreases upstream signaling
- Off signal
  - Sent when number of free buffers falls below threshold F_{off}
- On signal
  - Send when number of free buffers rises above threshold F_{on}

On-Off Timeline



- Less signaling but more buffering
  - On-chip buffers more expensive than wires

Virtual Channels vs. Physical Channels

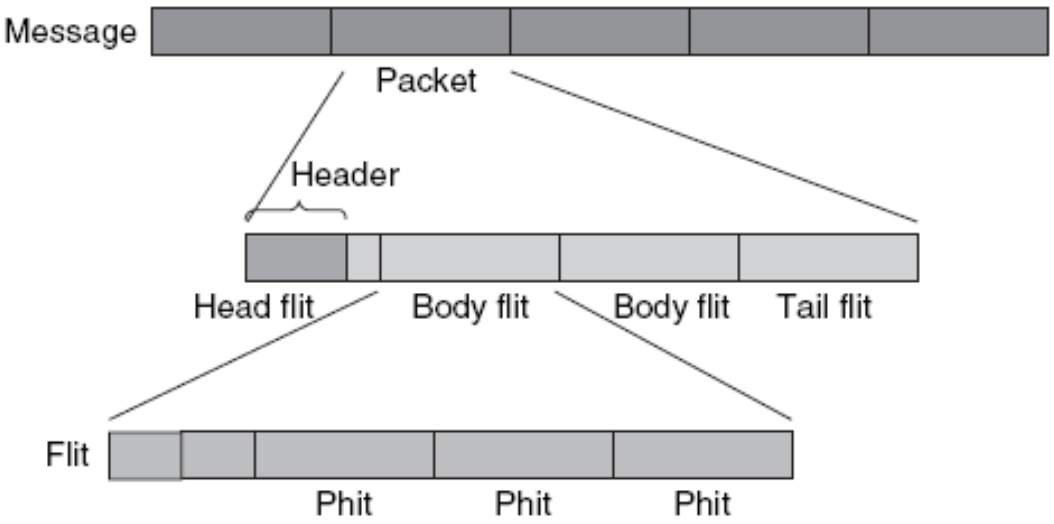
- Virtual channels share one physical set of links
  - Buffers hold flits that are in flight to free up physical channel
  - Flits from alternate VCs can traverse shared physical link
- This was the right design decision when:
  - Links were expensive (off-chip, backplane traces, or cables)
  - Buffers/router resources were cheap
  - Router latency was a small fraction of link latency
- With modern on-chip networks, this may not be true
  - Links are cheap (just global wires)
  - Buffer/router resources consume dynamic and static power, hence not cheap
  - Router latency is significant relative to link latency (usually just one cycle)
- Tilera design avoids virtual channels, simply provides multiple physical channels to avoid deadlock

Flow Control Summary

- On-chip networks require techniques with lower buffering requirements
  - Wormhole or Virtual Channel flow control
- Dropping packets unacceptable in on-chip environment
  - Requires buffer backpressure mechanism
- Complexity of flow control impacts router microarchitecture (next)

Switching strategies

- Determine how data flows through routers in the network
- Define granularity of data transfer and applied switching technique
  - phit is a unit of data that is transferred on a link in a single cycle
  - typically, phit size = flit size



Switching strategies

- Two main modes of transporting flits in a NoC are circuit switching and packet switching
- Circuit switching
  - physical path between the source and the destination is reserved prior to the transmission of data
  - message header flit traverses the network from the source to the destination, reserving links along the way
  - Advantage: low latency transfers, once path is reserved
  - Disadvantage: pure circuit switching does not scale well with NoC size
    - several links are occupied for the duration of the transmitted data, even when no data is being transmitted
      - for instance in the setup and tear down phases

Switching strategies

- Virtual circuit switching
  - creates virtual circuits that are multiplexed on links
  - number of virtual links (or virtual channels (VCs)) that can be supported by a physical link depends on buffers allocated to link
  - Possible to allocate either one buffer per virtual link or one buffer per physical link
  - Allocating one buffer per virtual link
    - depends on how virtual circuits are spatially distributed in the NoC, routers can have a different number of buffers
    - can be expensive due to the large number of shared buffers
    - multiplexing virtual circuits on a single link also requires scheduling at each router and link (end-to-end schedule)
    - conflicts between different schedules can make it difficult to achieve bandwidth and latency guarantees

Switching strategies

- Allocating one buffer per physical link
  - virtual circuits are time multiplexed with a single buffer per link
  - uses time division multiplexing (TDM) to statically schedule the usage of links among virtual circuits
  - flits are typically buffered at the NIs and sent into the NoC according to the TDM schedule
  - global scheduling with TDM makes it easier to achieve end-to-end bandwidth and latency guarantees
  - less expensive router implementation, with fewer buffers

Switching strategies

- Packet Switching
  - packets are transmitted from source and make their way independently to receiver
    - possibly along different routes and with different delays
  - zero start up time, followed by a variable delay due to contention in routers along packet path
  - QoS guarantees are harder to make in packet switching than in circuit switching
  - three main packet switching scheme variants
- SAF switching
  - packet is sent from one router to the next only if the receiving router has buffer space for entire packet
  - buffer size in the router is at least equal to the size of a packet
  - Disadvantage: excessive buffer requirements

Switching strategies

- VCT Switching
  - reduces router latency over SAF switching by forwarding first flit of a packet as soon as space for the entire packet is available in the next router
  - if no space is available in receiving buffer, no flits are sent, and the entire packet is buffered
  - same buffering requirements as SAF switching
- WH switching
  - flit from a packet is forwarded to receiving router if space exists for that flit
  - parts of the packet can be distributed among two or more routers
  - buffer requirements are reduced to one flit, instead of an entire packet
  - more susceptible to deadlocks due to usage dependencies between links

Outline

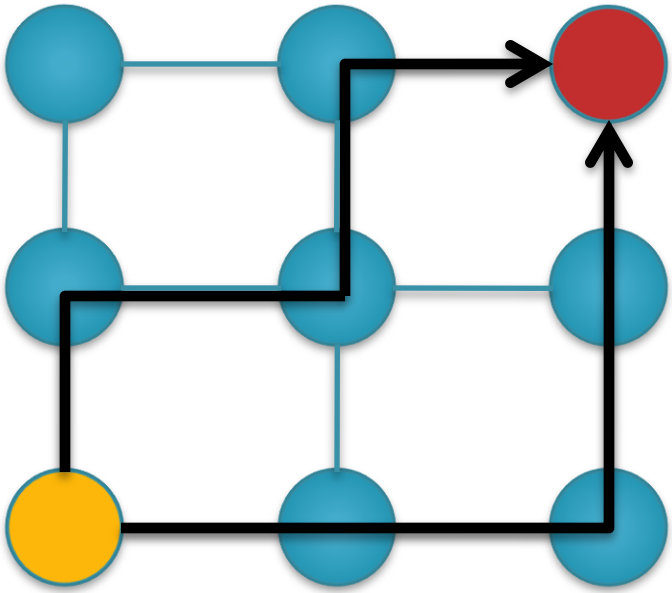
- Introduction
- NoC Topology
- Switching strategies
- Routing algorithms
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- Status and Open Problems

Routing Overview

- Discussion of topologies assumed ideal routing
- Practically though routing algorithms are not ideal
- Discuss various classes of routing algorithms
  - Deterministic, Oblivious, Adaptive
- Various implementation issues
  - Deadlock

Routing Basics

- Once topology is fixed
- Routing algorithm determines path(s) from source to destination



Routing Algorithm Attributes

- Number of destinations
  - Unicast, Multicast, Broadcast?
- Adaptivity
  - Oblivious or Adaptive? Local or Global knowledge?
- Implementation
  - Source or node routing?
  - Table or circuit?

Oblivious

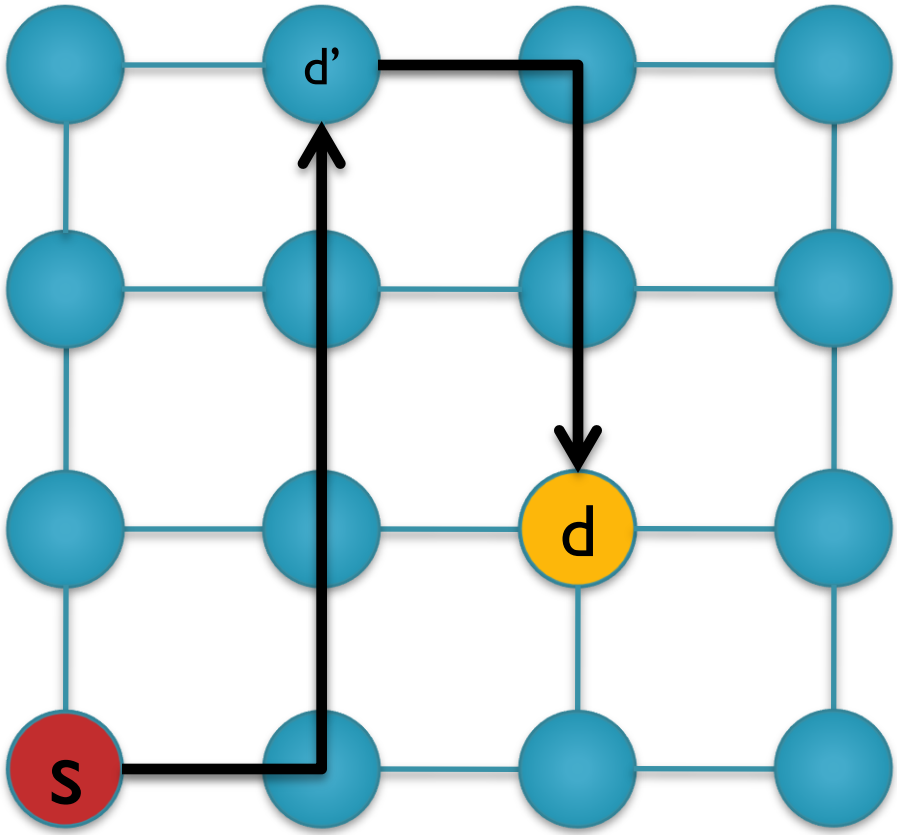
- Routing decisions are made without regard to network state
  - Keeps algorithms simple
  - Unable to adapt
- Deterministic algorithms are a subset of oblivious

Deterministic

- All messages from Src to Dest will traverse the same path
- Common example: Dimension Order Routing (DOR)
  - Message traverses network dimension by dimension
  - Aka XY routing
- Cons:
  - Eliminates any path diversity provided by topology
  - Poor load balancing
- Pros:
  - Simple and inexpensive to implement
  - Deadlock free

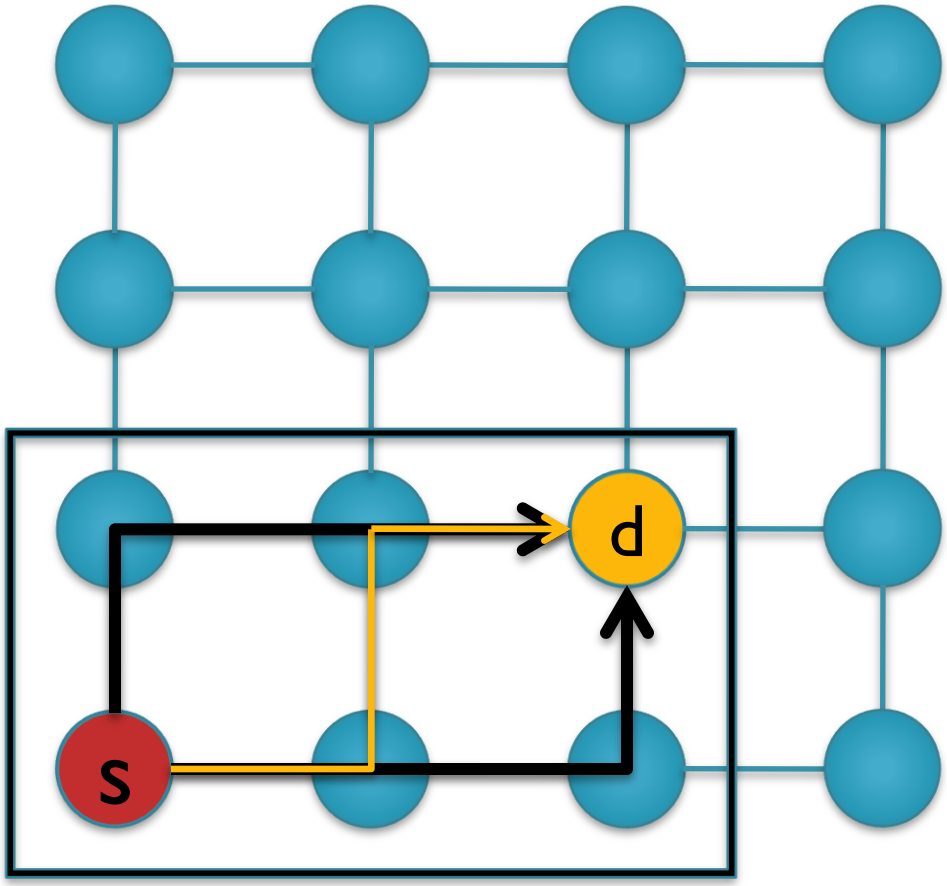
Valiant's Routing Algorithm

- To route from s to d, randomly choose intermediate node d'
  - Route from s to d' and from d' to d.
- Randomizes any traffic pattern
  - All patterns appear to be uniform random
  - Balances network load
- Non-minimal



Minimal Oblivious

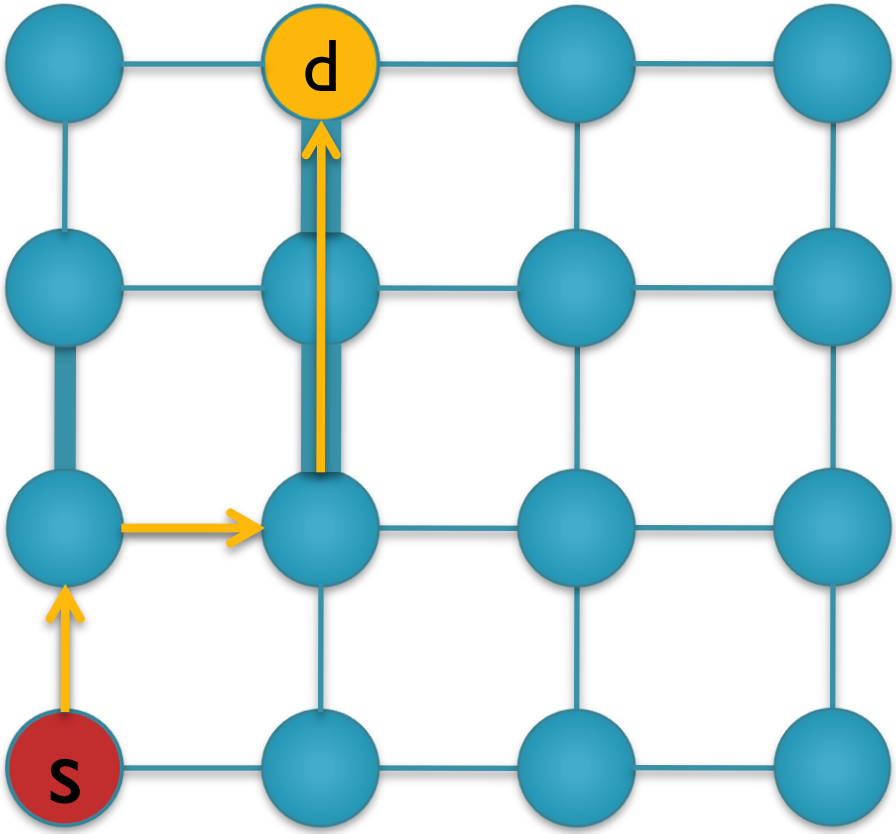
- Valiant's: Load balancing comes at expense of significant hop count increase
  - Destroys locality
- Minimal Oblivious: achieve some load balancing, but use shortest paths
  - d' must lie within minimum quadrant
  - 6 options for d'
  - Only 3 different paths



Adaptive

- Uses network state to make routing decisions
  - Buffer occupancies often used
  - Couple with flow control mechanism
- Local information readily available
  - Global information more costly to obtain
  - Network state can change rapidly
  - Use of local information can lead to non-optimal choices
- Can be minimal or non-minimal

Minimal Adaptive Routing

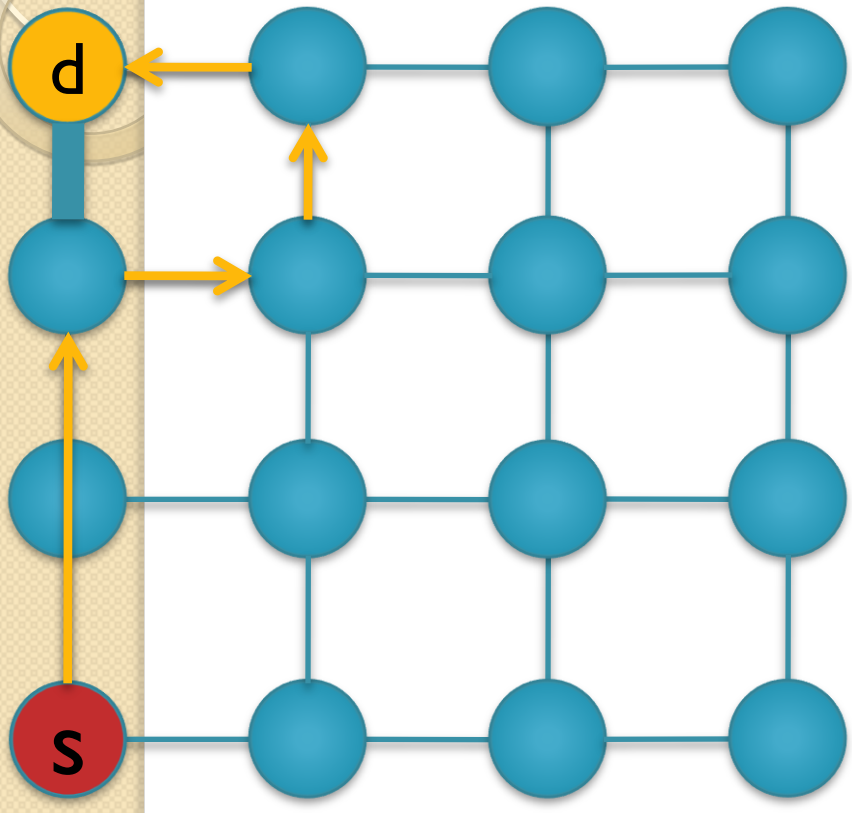


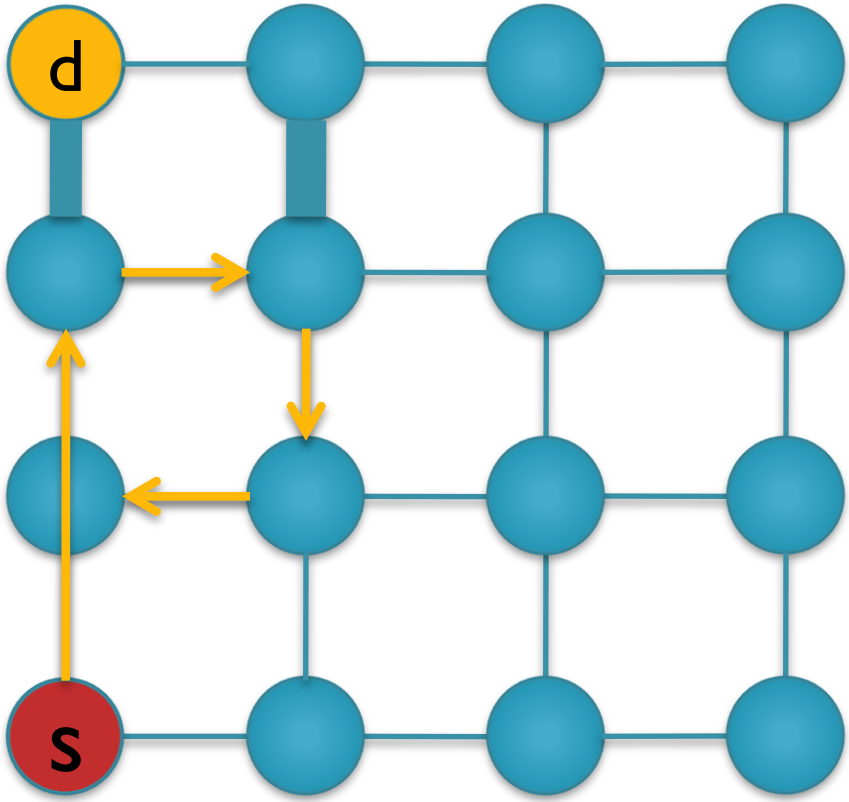
- Local info can result in sub-optimal choices

Non-minimal adaptive

- Fully adaptive
- Not restricted to take shortest path
  - Example: FBfly
- Misrouting: directing packet along non-productive channel
  - Priority given to productive output
  - Some algorithms forbid U-turns
- Livelock potential: traversing network without ever reaching destination
  - Mechanism to guarantee forward progress
    - Limit number of misroutings

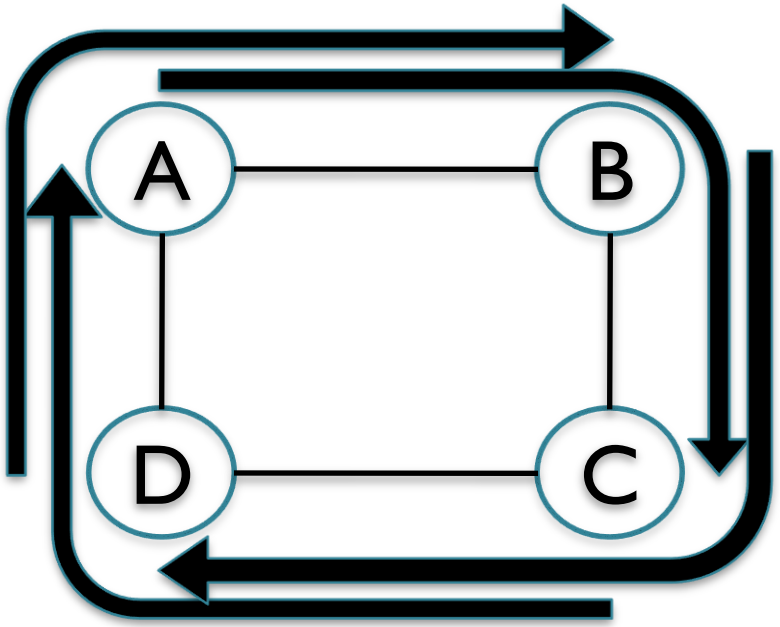
Non-minimal routing example





- Longer path with potentially lower latency
- Livelock: continue routing in cycle

Routing Deadlock

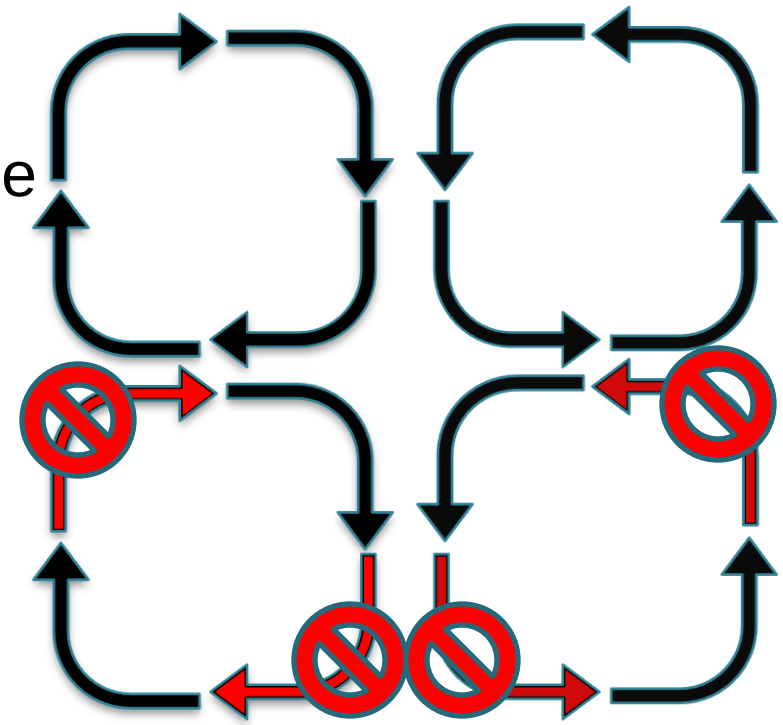


- Without routing restrictions, a resource cycle can occur
  - Leads to deadlock

Eliminate Cycles by Construction

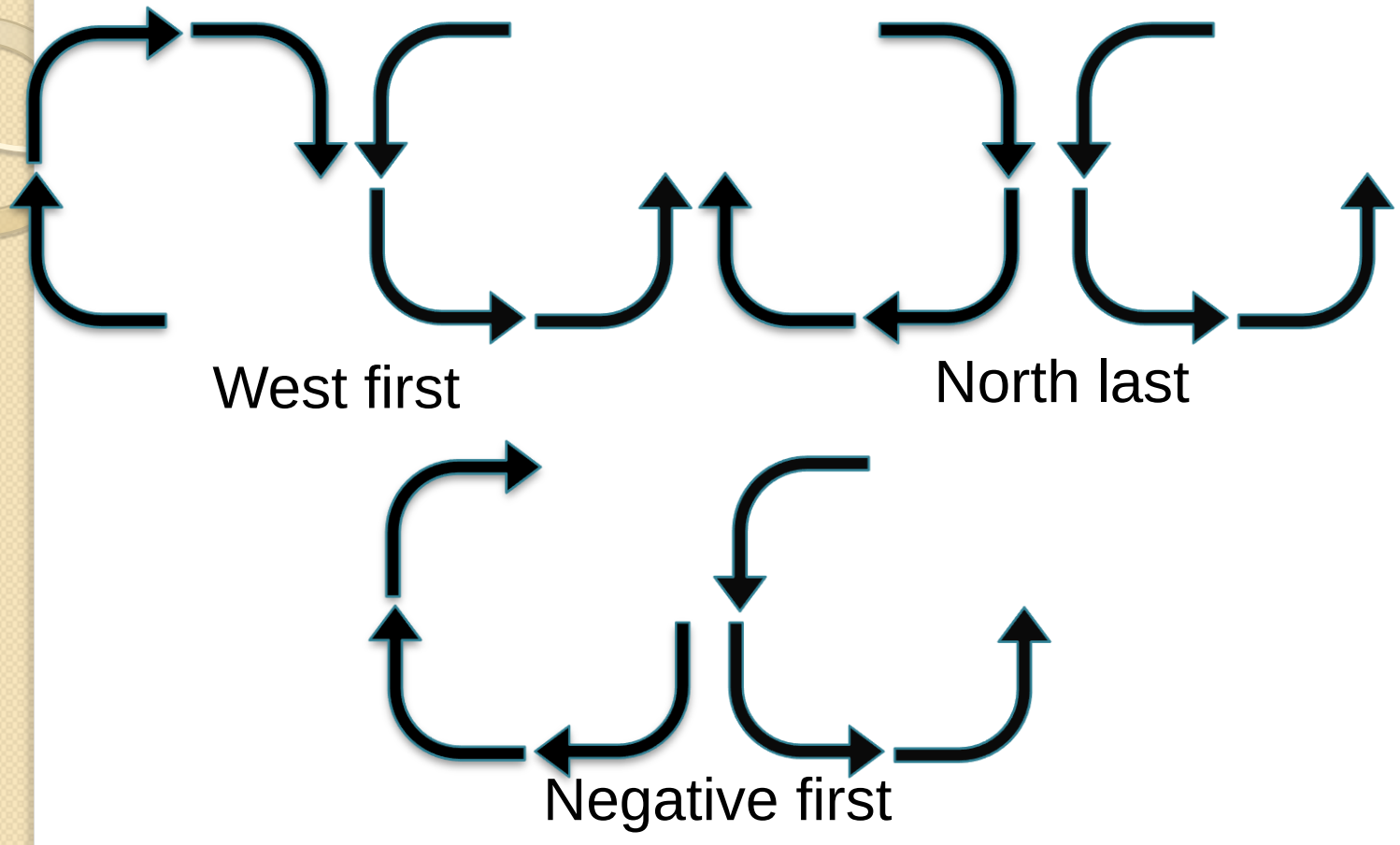
Deadlock Possible

XY Routing



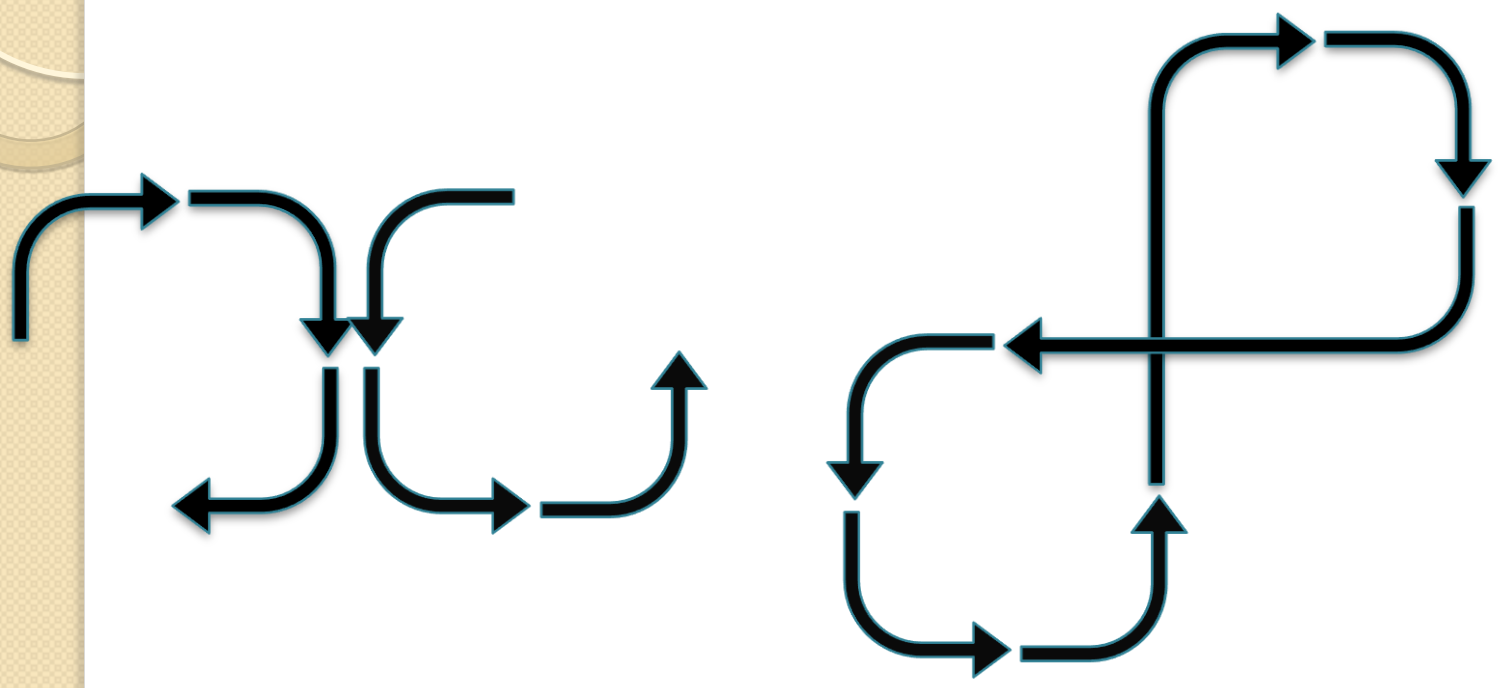
- Don't allow turns that cause cycles
- In general, acquire resources in fixed priority order

Turn Model Routing



- Some adaptivity by removing 2 of 8 turns
  - Remains deadlock free (but less restrictive than DOR)

Turn Model Routing Deadlock



- Not a valid turn elimination
  - Resource cycle results

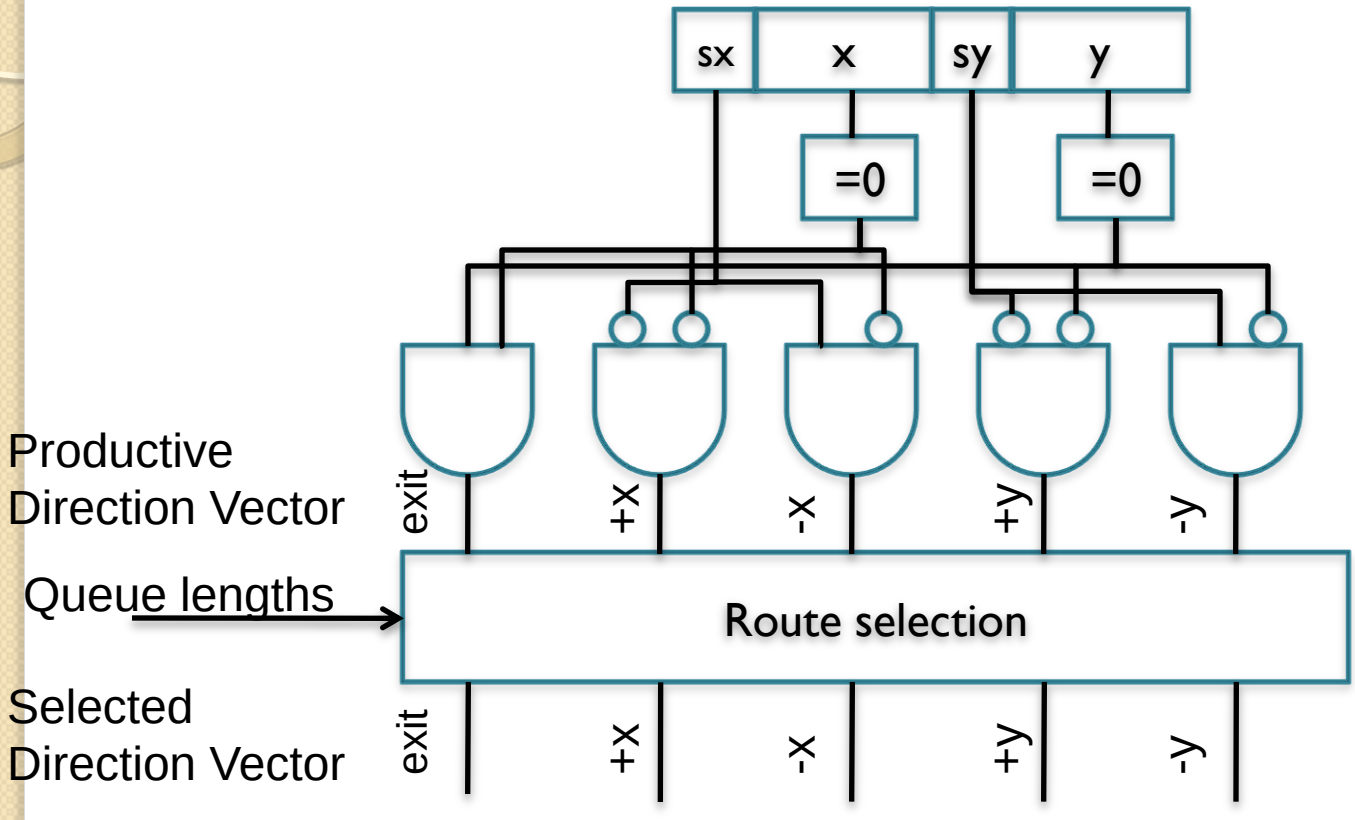
Routing Implementation

- Source tables
  - Entire route specified at source
  - Avoids per-hop routing latency
  - Unable to adapt to network conditions
  - Can specify multiple routes per destination
- Node tables
  - Store only next routes at each node
  - Smaller tables than source routing
  - Adds per-hop routing latency
  - Can adapt to network conditions
    - Specify multiple possible outputs per destination

Implementation

- Combinational circuits can be used
  - Simple (e.g. DOR): low router overhead
  - Specific to one topology and one routing algorithm
    - Limits fault tolerance
- Tables can be updated to reflect new configuration, network faults, etc

Circuit Based



Routing Summary

- Latency paramount concern
  - Minimal routing most common for NoC
  - Non-minimal can avoid congestion and deliver low latency
- To date: NoC research favors DOR for simplicity and deadlock freedom
  - On-chip networks often lightly loaded
- Only covered unicast routing
  - Recent work on extending on-chip routing to support multicast

Topology & Routing References

- Topology
  - William J. Dally and C. L Seitz. The torus routing chip. Journal of Distributed Computing, 1(3):187–196, 1986.
  - Charles Leiserson. Fat-trees: Universal networks for hardware efficient supercomputing. IEEE Transactions on Computers, 34(10), October 1985.
  - Boris Grot, Joel Hestness, Stephen W. Keckler, and OnurMutlu. Express cube topologies for on-chip networks. In Proceedings of the International Symposium on High Performance Computer Architecture, February 2009.
  - Flattened butterfly topology for on-chip networks. In Proceedings of the 40th International Symposium on Microarchitecture, December 2007.
  - J. Balfour and W. Dally. Design tradeoffs for tiled cmp on-chip networks. In Proceedings of the International Conference on Supercomputing, 2006.
- Routing
  - L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In Proceedings of the 13th Annual ACM Symposium on Theory of Computing, pages 263–277, 1981.
  - D. Seo, A. Ali, W.-T. Lim, N. Rafique, and M. Thottenhodi. Near-optimal worst- case throughput routing in two dimensional mesh networks. In Proceedings of the 32nd Annual International Symposium on Computer Architecture, June.
  - Christopher J. Glass and Lionel M. Ni. The turn model for adaptive routing. In Proceedings of the International Symposium on Computer Architecture, 1992.
  - P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in Proceedings of the 14th IEEE International Symposium on High-Performance Computer Architecture, February 2008.
  - N. EnrightJerger, L.-S. Peh, and M. H. Lipasti, "Virtual circuit tree multi- casting: A case for on-chip hardware multicast support," in Proceedings of the International Symposium on Computer Architecture (ISCA-35), Beijing, China, June 2008.

Routing algorithms

- Responsible for correctly and efficiently routing packets or circuits from the source to the destination
- Choice of a routing algorithm depends on trade-offs between several potentially conflicting metrics
  - minimizing power required for routing
  - minimizing logic and routing tables to achieve a lower area footprint
  - increasing performance by reducing delay and maximizing traffic utilization of the network
  - improving robustness to better adapt to changing traffic needs
- Routing schemes can be classified into several categories
  - static or dynamic routing
  - distributed or source routing
  - minimal or non-minimal routing

Routing algorithms

- Static and dynamic routing
  - static routing: fixed paths are used to transfer data between a particular source and destination
    - does not take into account current state of the network
  - advantages of static routing:
    - easy to implement, since very little additional router logic is required
    - in-order packet delivery if single path is used
  - dynamic routing: routing decisions are made according to the current state of the network
    - considering factors such as availability and load on links
  - path between source and destination may change over time
    - as traffic conditions and requirements of the application change
  - more resources needed to monitor state of the network and dynamically change routing paths
  - able to better distribute traffic in a network

Routing algorithms

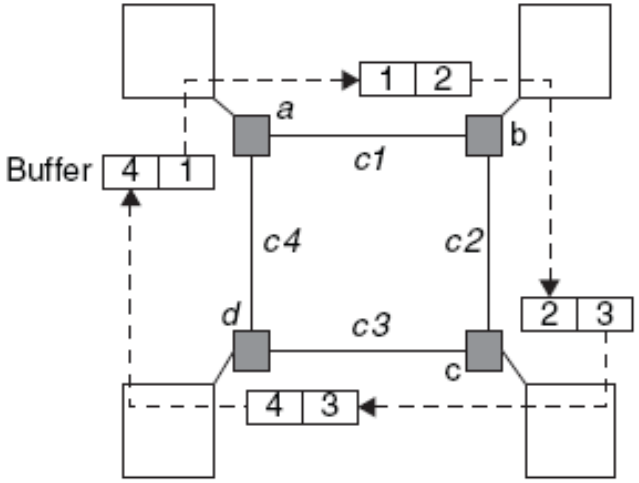
- Distributed and source routing
  - static and dynamic routing schemes can be further classified depending on where the routing information is stored, and where routing decisions are made
  - distributed routing: each packet carries the destination address
    - e.g., XY co-ordinates or number identifying destination node/router
    - routing decisions are made in each router by looking up the destination addresses in a routing table or by executing a hardware function
  - source routing: packet carries routing information
    - pre-computed routing tables are stored at a nodes' NI
    - routing information is looked up at the source NI and routing information is added to the header of the packet (increasing packet size)
    - when a packet arrives at a router, the routing information is extracted from the routing field in the packet header
    - does not require a destination address in a packet, any intermediate routing tables, or functions needed to calculate the route

Routing algorithms

- Minimal and non-minimal routing
  - minimal routing: length of the routing path from the source to the destination is the shortest possible length between the two nodes
    - e.g. in a mesh NoC topology (where each node can be identified by its XY co-ordinates in the grid) if source node is at (0, 0) and destination node is at (i, j), then the minimal path length is |i| + |j|
    - source does not start sending a packet if minimal path is not available
  - non-minimal routing: can use longer paths if a minimal path is not available
    - by allowing non-minimal paths, the number of alternative paths is increased, which can be useful for avoiding congestion
    - disadvantage: overhead of additional power consumption

Routing algorithms

- Routing algorithm must ensure freedom from deadlocks
  - common in WH switching
  - e.g. cyclic dependency shown below



  - freedom from deadlocks can be ensured by allocating additional hardware resources or imposing restrictions on the routing
  - usually dependency graph of the shared network resources is built and analyzed either statically or dynamically

Routing algorithms

- Routing algorithm must ensure freedom from livelocks
  - livelocks are similar to deadlocks, except that states of the resources involved constantly change with regard to one another, without making any progress
    - occurs especially when dynamic (adaptive) routing is used
    - e.g. can occur in a deflective "hot potato" routing if a packet is bounced around over and over again between routers and never reaches its destination
  - livelocks can be avoided with simple priority rules
- Routing algorithm must ensure freedom from starvation
  - under scenarios where certain packets are prioritized during routing, some of the low priority packets never reach their intended destination
  - can be avoided by using a fair routing algorithm, or reserving some bandwidth for low priority data packets

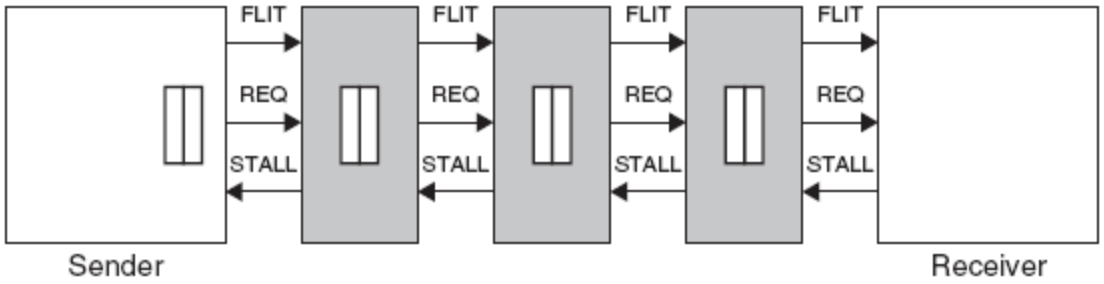
Outline

- Introduction
- NoC Topology
- Switching strategies
- Routing algorithms
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- Status and Open Problems

Flow control schemes

- Goal of flow control is to allocate network resources for packets traversing a NoC
  - can also be viewed as a problem of resolving contention during packet traversal
- At the data link-layer level, when transmission errors occur, recovery from the error depends on the support provided by the flow control mechanism
  - e.g. if a corrupted packet needs to be retransmitted, flow of packets from the sender must be stopped, and request signaling must be performed to reallocate buffer and bandwidth resources
- Most flow control techniques can manage link congestion
- But not all schemes can (by themselves) reallocate all the resources required for retransmission when errors occur
  - either error correction or a scheme to handle reliable transfers must be implemented at a higher layer

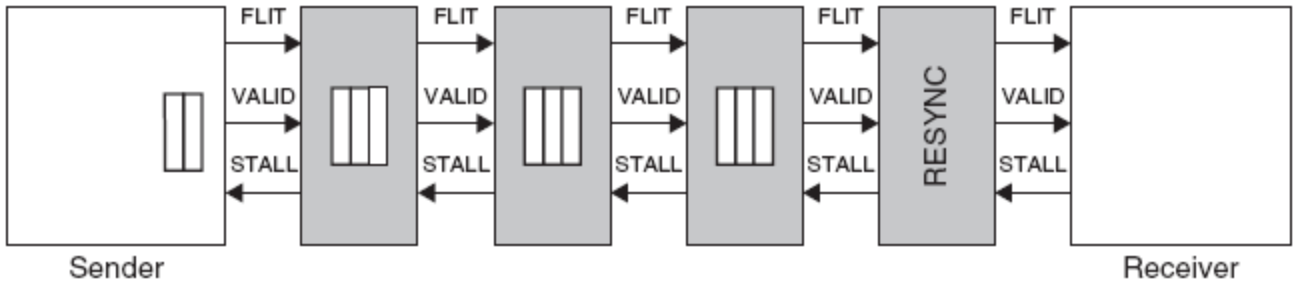
Flow control schemes



- STALL/GO
  - low overhead scheme
  - requires only two control wires
    - one going forward and signaling data availability
    - the other going backward and signaling either a condition of buffers filled (STALL) or of buffers free (GO)
  - can be implemented with distributed buffering (pipelining) along link
  - good performance – fast recovery from congestion
  - does not have any provision for fault handling
    - higher level protocols responsible for handling flit interruption

Flow control schemes

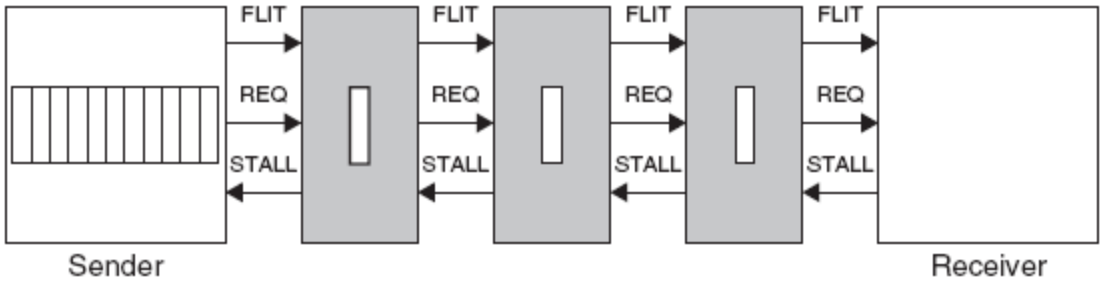
- T-Error



  - more aggressive scheme that can detect faults
    - by making use of a second delayed clock at every buffer stage
  - delayed clock re-samples input data to detect any inconsistencies
    - then emits a VALID control signal
  - resynchronization stage added between end of link and receiving switch
    - to handle offset between original and delayed clocks
  - timing budget can be used to provide greater reliability by configuring links with appropriate spacing and frequency
  - does not provide a thorough fault handling mechanism

Flow control schemes

- ACK/NACK



  - when flits are sent on a link, a local copy is kept in a buffer by sender
  - when ACK received by sender, it deletes copy of flit from its local buffer
  - when NACK is received, sender rewinds its output queue and starts resending flits, starting from the corrupted one
  - implemented either end-to-end or switch-to-switch
  - sender needs to have a buffer of size $2N + k$
    - N is number of buffers encountered between source and destination
    - k depends on latency of logic at the sender and receiver
  - overall a minimum of 3N + k buffers are required
  - fault handling support comes at cost of greater power, area overhead

Flow control schemes

- Network and Transport-Layer Flow Control
  - Flow Control without Resource Reservation
    - Technique #1: drop packets when receiver NI full
      - improves congestion in short term but increases it in long term
    - Technique #2: return packets that do not fit into receiver buffers to sender
      - to avoid deadlock, rejected packets must be accepted by sender
    - Technique #3: deflection routing
      - when packet cannot be accepted at receiver, it is sent back into network
      - packet does not go back to sender, but keeps hopping from router to router till it is accepted at receiver
  - Flow Control with Resource Reservation
    - credit-based flow control with resource reservation
    - credit counter at sender NI tracks free space available in receiver NI buffers
    - credit packets can piggyback on response packets
    - end-to-end or link-to-link

Outline

- Introduction
- NoC Topology
- Switching strategies
- Routing algorithms
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- Status and Open Problems

Clocking schemes

- Fully synchronous
  - single global clock is distributed to synchronize entire chip
    - hard to achieve in practice, due to process variations and clock skew
- Mesochronous
  - local clocks are derived from a global clock
  - not sensitive to clock skew
  - phase between clock signals in different modules may differ
    - deterministic for regular topologies (e.g. mesh)
    - non-deterministic for irregular topologies
      - synchronizers needed between clock domains
- Pleisochronous
  - clock signals are produced locally
- Asynchronous
  - clocks do not have to be present at all

Outline

- Introduction
- NoC Topology
- Switching strategies
- Routing algorithms
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- Status and Open Problems

Quality of Service (QoS)

- QoS refers to the level of commitment for packet delivery
  - refers to bounds on performance (bandwidth, delay, and jitter)
- Three basic categories
  - best effort (BE)
    - only correctness and completion of communication is guaranteed
    - usually packet switched
    - worst case times cannot be guaranteed
  - guaranteed service (GS)
    - makes a tangible guarantee on performance, in addition to basic guarantees of correctness and completion for communication
    - usually (virtual) circuit switched
  - differentiated service
    - prioritizes communication according to different categories
    - NoC switches employ priority based scheduling and allocation policies
    - cannot provide strong guarantees

Outline

- Introduction
- NoC Topology
- Switching strategies
- Routing algorithms
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- Status and Open Problems

Æthereal

- Developed by Philips
- Synchronous indirect network
- WH switching
- Contention-free source routing based on TDM
- GT as well as BE QoS
- GT slots can be allocated statically at initialization phase, or dynamically at runtime
- BE traffic makes use of non-reserved slots, and any unused reserved slots
  - also used to program GT slots of the routers
- Link-to-link credit-based flow control scheme between BE buffers
  - to avoid loss of flits due to buffer overflow

HERMES

- Developed at the Faculdade de Informática PUCRS, Brazil
- Direct network
- 2-D mesh topology
- WH switching with minimal XY routing algorithm
- 8 bit flit size; first 2 flits of packet contain header
- Header has target address and number of flits in the packet
- Parameterizable input queuing
  - to reduce the number of switches affected by a blocked packet
- Connectionless: cannot provide any form of bandwidth or latency GS

MANGO

- Message-passing Asynchronous Network-on-chip providing GS over open core protocol (OCP) interfaces
- Developed at the Technical University of Denmark
- Clockless NoC that provides BE as well as GS services
- NIs (or adapters) convert between the synchronous OCP domain and asynchronous domain
- Routers allocate separate physical buffers for VCs
  - For simplicity, when ensuring GS
- BE connections are source routed
  - BE router uses credit-based buffers to handle flow control
  - length of a BE path is limited to five hops
- static scheduler gives link access to higher priority channels
  - admission controller ensures low priority channels do not starve

Nostrum

- Developed at KTH in Stockholm
- Direct network with a 2-D mesh topology
- SAF switching with hot potato (or deflective) routing
- Support for
  - switch/router load distribution
  - guaranteed bandwidth (GB)
  - multicasting
- GB is realized using looped containers
  - implemented by VCs using a TDM mechanism
  - container is a special type of packet which loops around VC
  - multicast: simply have container loop around on VC having recipients
- Switch load distribution requires each switch to indicate its current load by sending a stress value to its neighbors

Octagon

- Developed by STMicroelectronics
- direct network with an octagonal topology
- 8 nodes and 12 bidirectional links
- Any node can reach any other node with a max of 2 hops
- Can operate in packet switched or circuit switched mode
- Nodes route a packet in packet switched mode according to its destination field
  - node calculates a relative address and then packet is routed either left, right, across, or into the node
- Can be scaled if more than 8 nodes are required
  - Spidergon

QNoC

- Developed at Technion in Israel
- Direct network with an irregular mesh topology
- WH switching with an XY minimal routing scheme
- Link-to-link credit-based flow control
- Traffic is divided into four different service classes
  - signaling, real-time, read/write, and block-transfer
  - signaling has highest priority and block transfers lowest priority
  - every service level has its own small buffer (few flits) at switch input
- Packet forwarding is interleaved according to QoS rules
  - high priority packets able to preempt low priority packets
- Hard guarantees not possible due to absence of circuit switching
  - Instead statistical guarantees are provided

SOCBUS

- Developed at Linköping University
- Mesochronous clocking with signal retiming is used
- Circuit switched, direct network with 2-D mesh topology
- Minimum path length routing scheme is used
- Circuit switched scheme is
  - deadlock free
  - requires simple routing hardware
  - very little buffering (only for the request phase)
  - results in low latency
- Hard guarantees are difficult to give because it takes a long time to set up a connection

SPIN

- Scalable programmable integrated network (SPIN)
- fat-tree topology, with two one-way 32-bit link data paths
- WH switching, and deflection routing
- Virtual socket interface alliance (VSIA) virtual component interface (VCI) protocol to interface between PEs
- Flits of size 4 bytes
- First flit of packet is header
  - first byte has destination address (max. 256 nodes)
  - last byte has checksum
- Link level flow control
- Random hiccups can be expected under high load
  - GS is not supported

Xpipes

- Developed by the Univ. of Bologna and Stanford University
- Source-based routing, WH switching
- Supports OCP standard for interfacing nodes with NoC
- Supports design of heterogeneous, customized (possibly irregular) network topologies
- go-back-N retransmission strategy for link level error control
  - errors detected by a CRC (cycle redundancy check) block running concurrently with the switch operation
- XpipesCompiler and NetChip compilers
  - Tools to tune parameters such as flit size, address space of cores, max. number of hops between any two network nodes, etc.
  - generate various topologies such as mesh, torus, hypercube, Clos, and butterfly

Outline

- Introduction
- NoC Topology
- Switching strategies
- Routing algorithms
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- Status and Open Problems

Status and Open Problems

- Power
  - complex NI and switching/routing logic blocks are power hungry
  - several times greater than for current bus-based approaches
- Latency
  - additional delay to packetize/de-packetize data at NIs
  - flow/congestion control and fault tolerance protocol overheads
  - delays at the numerous switching stages encountered by packets
  - even circuit switching has overhead (e.g. SOCBUS)
  - lags behind what can be achieved with bus-based/dedicated wiring
- Lack of tools and benchmarks
- Simulation speed
  - GHz clock frequencies, large network complexity, greater number of PEs slow down simulation

Trends

- Move towards hybrid interconnection fabrics
  - NoC-bus based
  - Custom, heterogeneous topologies
- New interconnect paradigms
  - Optical
  - Wireless
  - Carbon nanotube

