Interconnection Networks
Interconnection Networks
Introduction

• How to connect individual devices together into a group of communicating devices?

• Device:
  ❑ Component within a computer
  ❑ Single computer
  ❑ System of computers

• Types of elements:
  ❑ end nodes (device + interface)
  ❑ links
  ❑ interconnection network

• Internetworking: interconnection of multiple networks
Interconnection Networks Introduction

- **Interconnection networks** should be designed
  - to transfer the **maximum amount of information**
  - within the **least amount of time** (and cost, power constraints)
  - so as not to bottleneck the system
Types of Interconnection Networks

• Four different domains:
  - Depending on number & proximity of connected devices

• On-Chip networks (OCNs or NoCs)
  - Devices are microarchitectural elements (functional units, register files), caches, directories, processors
  - Latest systems: dozens, hundreds of devices
    - Ex: Intel TeraFLOPS research prototypes – 80 cores
    - Xeon Phi – 60 cores
  - Proximity: millimeters
System/Storage Area Networks (SANs)

- Multiprocessor and multicore systems
  - Interprocessor and processor-memory interconnections

- Server and data center environments
  - Storage and I/O components

- Hundreds to thousands of devices interconnected
  - IBM Blue Gene/L supercomputer
    - (64K nodes, each with 2 processors)

- Maximum interconnect distance
  - Tens of meters (typical)
  - A few hundred meters (some)
    - InfiniBand: 120 Gbps over a distance of 300m

- Examples (standards and proprietary)
  - InfiniBand, Myrinet, Quadrics, Advanced Switching Interconnect
Local Area Network (LANs)

- Interconnect autonomous computer systems
- Machine room or throughout a building or campus
- Hundreds of devices interconnected (1,000s with bridging)
- Maximum interconnect distance
  - few kilometers
  - few tens of kilometers (some)
- Example (most popular): Ethernet, with 10 Gbps over 40Km
Wide Area Networks (WANs)

- Interconnect systems distributed across the globe
- Internetworking support is required
- Many millions of devices interconnected
- Maximum interconnect distance
  - many thousands of kilometers
- Example: ATM (asynchronous transfer mode)
Interconnection Network Domains

Distance (meters)

- $5 \times 10^{-3}$
- $5 \times 10^{0}$
- $5 \times 10^{3}$
- $5 \times 10^{6}$

Number of devices interconnected

- $1$
- $10$
- $100$
- $1,000$
- $10,000$
- $>100,000$

OCNs

SANs

LANs

WANs
EECS 570 Focus: On-Chip Networks
On-Chip Networks (OCN or NoCs)

• Why On-Chip Network?
  - Ad-hoc wiring does not scale beyond a small number of cores
    - Prohibitive area
    - Long latency

• OCN offers
  - scalability
  - efficient multiplexing of communication
  - often modular in nature (eases verification)
Differences between on-chip and off-chip networks

- Significant research in multi-chassis interconnection networks (off-chip)
  - Supercomputers
  - Clusters of workstations
  - Internet routers

- Leverage research and insight but...
  - Constraints are different
Off-chip vs. on-chip

• Off-chip: I/O bottlenecks
  - Pin-limited bandwidth
  - Inherent overheads of off-chip I/O transmission

• On-chip
  - Wiring constraints
    - Metal layer limitations
    - Horizontal and vertical layout
    - Short, fixed length
    - Repeater insertion limits routing of wires
      - Avoid routing over dense logic
      - Impact wiring density
  - Power
    - Consume 10-15% or more of die power budget
  - Latency
    - Different order of magnitude
    - Routers consume significant fraction of latency
On-Chip Network Evolution

- Ad hoc wiring
  - Small number of nodes

- Buses and Crossbars
  - Simplest variant of on-chip networks
  - Low core counts
  - Like traditional multiprocessors
    - Bus traffic quickly saturates with a modest number of cores
  - Crossbars: higher bandwidth
    - Poor area and power scaling
Multicore Examples (1)

Sun Niagara

- Niagara 2: 8x9 crossbar (area ~ core)
- Rock: Hierarchical crossbar (5x5 crossbar connecting clusters of 4 cores)
Multicore Examples (2)

- IBM Cell
- Element Interconnect Bus
  - 12 elements
  - 4 unidirectional rings
    - 16 Bytes wide
    - Operates at 1.6 GHz
Many Core Example

- Intel TeraFLOPS
  - 80 core prototype
  - 5 GHz
  - Each tile:
    - Processing engine + on-chip network router

<table>
<thead>
<tr>
<th>Technology</th>
<th>65nm CMOS Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interconnect</td>
<td>1 poly, 8 metal (Cu)</td>
</tr>
<tr>
<td>Transistors</td>
<td>100 Million</td>
</tr>
<tr>
<td>Die Area</td>
<td>275mm$^2$</td>
</tr>
<tr>
<td>Tile area</td>
<td>3mm$^2$</td>
</tr>
<tr>
<td>Package</td>
<td>1248 pin LGA, 14 layers, 343 signal pins</td>
</tr>
</tbody>
</table>
Many-Core Example (2): Intel SCC

- Intel’s Single-chip Cloud Computer (SCC) uses a 2D mesh with state of the art routers
Performance and Cost

- Performance: latency and throughput
- Cost: area and power

Latency (sec)

Offered Traffic (bits/sec)

Zero load latency

Saturation throughput
Topics to be covered

- Interfaces
- Topology
- Routing
- Flow Control
- Router Microarchitecture
System Interfaces
Systems and Interfaces

• Look at how systems interact and interface with network

• Types of multi-processors
  - Shared-memory
    ◉ From high end servers to embedded products
  - Message passing
    ◉ Multiprocessor System on Chip (MPSoC)
      ◉ Mobile consumer market
    ◉ Clusters

• We focus on on-chip networks for shared-memory multi-core
Shared Memory CMP Architecture

L2:
- Private or distributed shared cache
- Centralized shared cache will have a different organization

A tile could be a core or L2 bank
Impact of Coherence Protocol on Network Performance

- Coherence protocol shapes communication needed by system

- Single writer, multiple reader invariant

- Requires:
  - Data requests
  - Data responses
  - Coherence permissions
Broadcast vs. Directory

1. Read Cache miss
2. Send Data
3. Request broadcast

Directory receives request
2. Directory
3. Send Data

Read Cache miss
Coherence Protocol Requirements

- Different message types
  - Unicast, multicast, broadcast

- Directory protocol
  - Majority of requests: Unicast
    - Lower bandwidth demands on network
  - More scalable due to point-to-point communication

- Broadcast protocol
  - Majority of requests: Broadcast
    - Higher bandwidth demands
  - Often rely on network ordering
Protocol Level Deadlock

• Request-Reply Dependency
  - Network becomes flooded with requests that cannot be consumed until the network interface has generated a reply

• Deadlock dependency between multiple message classes

• Virtual channels can prevent protocol level deadlock (to be discussed later)
Home Node/Memory Controller Issues

- Heterogeneity in network
  - Some tiles are memory controllers
    - Co-located with processor/cache or separate tile
    - Share injection/ejection bandwidth?

- Home node
  - Directory coherence information
  - <= number of tiles

- Potential hot spots in network?
Network Interface
Network Interface: Miss Status Handling Registers

- Core
  - Cache
    - Protocol Finite State Machine
      - MSHRs
        - Status
          - Addr
          - Data
        - Type
        - Addr
        - Data
    - Message Format and Send
      - To network
        - Dest
        - RdReq
        - Addr
    - Message Receive
      - From network
        - RdReply
        - Addr
        - Data
        - Request
        - Addr
        - WriteAck
        - Addr
        - Reply
        - Addr
        - Data
Transaction Status Handling Registers
(for centralized directory)

<table>
<thead>
<tr>
<th>Src</th>
<th>RdReq</th>
<th>Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Src</th>
<th>Writeback</th>
<th>Addr</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dest</th>
<th>RdReply</th>
<th>Addr</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dest</th>
<th>WriteAck</th>
<th>Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From network

Message Receive

Directory Cache

To network

Message Format and Send

Directory Cache

Memory Controller

Off-chip memory

TSHRs

<table>
<thead>
<tr>
<th>Status</th>
<th>Src</th>
<th>Addr</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From network

Message Receive

Directory Cache

To network

Message Format and Send

Directory Cache

Memory Controller

Off-chip memory

TSHRs
MPSoCs
Synthesized NoCs for MPSoCs

- System-on-Chip (SoC)
  - Chips tailored to specific applications or domains
  - Designed quickly through composition of IP blocks
- Fundamental NoC concepts applicable to both CMP and MPSoC
- Key characteristics
  - Applications known a priori
  - Automated design process
  - Standardized interfaces
  - Area/power constraints tighter
Application Characterization

- Describe application with task graphs
- Annotate with traffic volumes
Design Requirements

- Less aggressive
  - CMPs: GHz clock frequencies
  - MPSoCs: MHz clock frequencies
  - Pipelining may not be necessary
  - Standardizes interfaces add significant delay

- Area and power
  - CMPs: 100W for server
  - MPSoC: several watts only

- Time to market
  - Automatic composition and generation
NoC Synthesis

User objectives: power, hop delay
Constraints: area, power, hop delay, wire length

Topology Synthesis
Includes: Floorplanner NoC Router

System specs:
Platform Generation (xpipes-Compiler)

NoC Component library
IP Core models

FPGA Emulation
Synthesis
Placement and Routing
To fab

Floorplanning specifications
Area, power characterization

Input traffic model
Constraint graph Comm graph
NoC Area models
NoC Power models

Application

Codesign simulation

EECS 570
Lecture 19
Slide 35
NoC Synthesis

• Tool chain
  - Requires accurate power and area models
  - Quickly iterate through many designs
  - Library of soft macros for all NoC building blocks
  - Floorplanner
    - Determine router locations
    - Determine link lengths (delay)
NoC Network Interface Standards

- Standardized protocols
  - Plug and play with different IP blocks

- Bus-based semantics
  - Widely used

- Out of order transactions
  - Relax strict bus ordering semantics
  - Migrating MPSoCs from buses to NoCs.
Summary

- Architecture
  - Impacts communication requirements
  - Coherence protocol: Broadcast vs. Directory
  - Shared vs. Private Caches

- CMP vs. MPSoC
  - General vs. Application specific
  - Custom interfaces vs. standardized interfaces
Topics to be covered

• Interfaces
• Topology
• Routing
• Flow Control
• Router Microarchitecture
Types of Topologies
Types of Topologies

- Focus on switched topologies
  - Alternatives: bus and crossbar

- Bus
  - Connects a set of components to a single shared channel
  - Effective broadcast medium

- Crossbar
  - Directly connects $n$ inputs to $m$ outputs without intermediate stages
  - Fully connected, single hop network
  - Component of routers
Types of Topologies

• **Direct**
  - Each router is associated with a terminal node
  - *All* routers are sources and destinations of traffic

• **Indirect**
  - Routers are distinct from terminal nodes
  - Terminal nodes can source/sink traffic
  - Intermediate nodes switch traffic between terminal nodes

• Most on-chip network use direct topologies
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)

- 1D or 2D torus map well to planar substrate for on-chip
- Topologies in Torus Family
  - Ex: Ring -- k-ary 1-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
Torus (3)

- Degree = 2n, 2 channels per dimension
  - All nodes have same degree

- Total channels = 2nN
  - N is total number of nodes
Mesh

• A torus with end-around connection removed

• Same node degree

• Higher demand for central channels
  ☐ Load imbalance
Butterfly

• Indirect network

• K-ary n-fly: $k^n$ network nodes

• Routing from 000 to 010
  - Dest address used to directly route packet
  - Bit $n$ used to select output port at stage $n$

2-ary 3-fly
2 input switch, 3 stages
Butterfly (2)

- No path diversity \( |R_{xy}| = 1 \)
  - Can add extra stages for diversity
  - Increase network diameter
Butterfly (3)

- Hop Count
  - $\log_k N + 1$
  - Does not exploit locality
    - Hop count same regardless of location

- Switch Degree = 2k

- Requires long wires to implement
Clos network

- 3-stage networks where all input/output nodes are connected to all middle routers
- Key attribute: path diversity
  - Input node can select any middle router
  - Can enable non-blocking routing algorithms

(5,3,4) Clos network
Fat Tree

- Bandwidth remains constant at each level
- Regular Tree: Bandwidth decreases closer to root
Fat Tree (2)

- Provides path diversity
Irregular Topologies
Irregular Topologies

• MPSoC design leverages wide variety of IP blocks
  - Regular topologies may not be appropriate given heterogeneity
  - Customized topology
    - Often more power efficient and deliver better performance

• Customize based on traffic characterization
Irregular Topology Example

VLD
Run length decoder
Inverse scan
iDCT
iQuant
AC/DC predict
up samp
VOP reconstr
Stripe Memory
ARM core
VOP Memory
Padding
VLD
Run length decoder
Inverse scan
iDCT
iQuant
AC/DC predict
up samp
VOP reconstr
Stripe Memory
ARM core
VOP Memory
Padding
Topology Customization

• Merging
  - Start with large number of switches
  - Merge to adjacent routers reduce area and power

• Splitting
  - Large crossbar connecting all nodes
  - Iteratively split into multiple small switches
    - Accommodate design constraints
Topology Implementation
Implementation

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

• Don’t need 1:1 ratio of routers to cores
  ☐ Ex: 4 cores concentrated to 1 router

• Can save area and power

• Increases network complexity
  ☐ Concentrator must implement policy for sharing injection bandwidth
  ☒ During bursty communication
    ☒ Can bottleneck
Implication of Abstract Metrics on Implementation

- Degree: useful proxy for router complexity
  - Increasing ports requires additional buffer queues, requestors to allocators, ports to crossbar

- All contribute to critical path delay, area and power

- Link complexity does not correlate with degree
  - Link complexity depends on link width
  - Fixed number of wires, link complexity for 2-port vs 3-port is same
Implications (2)

- Hop Count: useful proxy for overall latency and power

  - Does not always correlate with latency
    - Depends heavily on router pipeline and link propagation

  - Example:
    - Network A with 2 hops, 5 stage pipeline, 4 cycle link traversal vs.
    - Network B with 3 hops, 1 stage pipeline, 1 cycle link traversal

  Hop Count says A is better than B
  But A has 18 cycle latency vs 6 cycle latency for B
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
Routing
Routing Overview

• Discussion of topologies assumed ideal routing

• In practice...
  - Routing algorithms are not ideal

• Goal: distribute traffic **evenly** among paths
  - Avoid hot spots, contention
  - More balanced → closer throughput is to ideal

• Keep complexity in mind
Routing Basics

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

- Types
  - Deterministic, Oblivious, Adaptive

- Number of destinations
  - Unicast, Multicast, Broadcast?

- Adaptivity
  - Oblivious or Adaptive? Local or Global knowledge?
  - Minimal or non-minimal?

- Implementation
  - Source or node routing?
  - Table or circuit?
Routing Deadlock

• Each packet is occupying a link and waiting for a link
• Without routing restrictions, a resource cycle can occur
  ☐ Leads to deadlock
Types of Routing Algorithms
Deterministic

- All messages from Source to Destination 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
Dimension Order Routing

• a.k.a X-Y Routing
  - Traverse network dimension by dimension
  - Can only turn to Y dimension after finished X
Oblivious

- Routing decisions are made without regard to network state
  - Keeps algorithms simple
  - Unable to adapt

- Deterministic algorithms are a subset of oblivious
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 uniform random
  - Balances network load

• Non-minimal

• Destroys locality
**Minimal Oblivious**

- Valiant’s: Load balancing but significant increase in hop count

- Minimal Oblivious: some load balancing, but use shortest paths
  - $d'$ must lie within min quadrant
  - 6 options for $d'$
  - Only 3 different paths
Oblivious Routing

- Valiant’s and Minimal Adaptive
  - Deadlock free
    - When used in conjunction with X-Y routing

- Randomly choose between X-Y and Y-X routes
  - Oblivious but not deadlock free!
Adaptive

• Exploits path diversity

• Uses network state to make routing decisions
  - Buffer occupancies often used
  - Coupled 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

• 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
Adaptive Routing Example

• Should 3 route clockwise or counterclockwise to 7?
  • 5 is using all the capacity of link 5 → 6

• Queue at node 5 will sense contention but not at node 3

• Backpressure: allows nodes to indirectly sense congestion
  • Queue in one node fills up, it will stop receiving flits
  • Previous queue will fill up

• If each queue holds 4 packets
  • 3 will send 8 packets before sensing congestion
Adaptive Routing: Turn Model

- DOR eliminates 4 turns
  - N to E, N to W, S to E, S to W
  - No adaptivity
- Some adaptivity by removing 2 of 8 turns
  - Remains deadlock free (like DOR)
- West first
  - Eliminates S to W and N to W

West first

Diagram:

- Arrows indicating directions and turn eliminations.
- "West first" label.
Turn Model Routing

- Negative first
  - Eliminates E to S and N to W

- North last
  - Eliminates N to E and N to W

- Odd-Even
  - Eliminates 2 turns depending on if current node is in odd or even col.
    - Even column: E to N and N to W
    - Odd column: E to S and S to W
  - Deadlock free (disallow 180 turns)
  - Better adaptivity
Negative-First Routing Example

- Limited or no adaptivity for certain source-destination pairs
Turn Model Routing Deadlock

• What about eliminating turns NW and WN?
• Not a valid turn elimination
  - Resource cycle results
Adaptive Routing and Deadlock

Option 1: Eliminate turns that lead to deadlock
- Limits flexibility

Option 2: Allow all turns
- Give more flexibility
- Must use other mechanism to prevent deadlock
- Rely on flow control (later)
  - Escape virtual channels
Routing Algorithm Implementation
Routing Implementation

- Source tables
  - Entire route specified at source
  - Avoids per-hop routing latency
  - Unable to adapt dynamically to network conditions
  - Can specify multiple routes per destination
    - Give fault tolerance and load balance
  - Support reconfiguration (not specific to topology)
## Source Table Routing

<table>
<thead>
<tr>
<th>Destination</th>
<th>Route 1</th>
<th>Route 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td>EX</td>
<td>EX</td>
</tr>
<tr>
<td>20</td>
<td>EEX</td>
<td>EEX</td>
</tr>
<tr>
<td>01</td>
<td>NX</td>
<td>NX</td>
</tr>
<tr>
<td>11</td>
<td>NEX</td>
<td>ENX</td>
</tr>
<tr>
<td>21</td>
<td>NEEX</td>
<td>ENEX</td>
</tr>
<tr>
<td>02</td>
<td>NNX</td>
<td>NNX</td>
</tr>
<tr>
<td>12</td>
<td>ENNX</td>
<td>NNEX</td>
</tr>
<tr>
<td>22</td>
<td>EENNX</td>
<td>NNEEX</td>
</tr>
<tr>
<td>03</td>
<td>NNNX</td>
<td>NNNX</td>
</tr>
<tr>
<td>13</td>
<td>NENNX</td>
<td>ENNNX</td>
</tr>
<tr>
<td>23</td>
<td>EENNNX</td>
<td>NNNEEX</td>
</tr>
</tbody>
</table>

- Arbitrary length paths: storage overhead and packet overhead
Node Tables

• Store only next direction at each node

• Smaller tables than source routing

• Adds per-hop routing latency

• Can adapt to network conditions
  ◆ Specify multiple possible outputs per destination
  ◆ Select randomly to improve load balancing
Node Table Routing

<table>
<thead>
<tr>
<th>From</th>
<th>00</th>
<th>01</th>
<th>02</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>X</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>E</td>
<td>-</td>
<td>E</td>
</tr>
<tr>
<td>01</td>
<td>S</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>N</td>
<td>-</td>
<td>E</td>
<td>S</td>
<td>E</td>
</tr>
<tr>
<td>02</td>
<td>S</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>E</td>
<td>S</td>
<td>E</td>
<td>S</td>
<td>E</td>
</tr>
<tr>
<td>10</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>N</td>
</tr>
<tr>
<td>11</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>S</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>12</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>S</td>
<td>-</td>
<td>S</td>
</tr>
<tr>
<td>20</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>W</td>
</tr>
<tr>
<td>21</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>W</td>
</tr>
<tr>
<td>22</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>W</td>
<td>-</td>
<td>W</td>
</tr>
</tbody>
</table>

- Each node would have 1 row of table
  - Max two possible output ports
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

- Next hop based on buffer occupancies
- Or could implement simple DOR
- Fixed w.r.t. topology
## Routing Algorithms: Implementation

<table>
<thead>
<tr>
<th>Routing Algorithm</th>
<th>Source Routing</th>
<th>Combinational</th>
<th>Node Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOR</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Oblivious</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valiant’s</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Minimal</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Adaptive</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Routing: Irregular Topologies

- MPSoCs
  - Power and performance benefits from irregular/custom topologies

- Common routing implementations
  - Rely on source or node table routing

- Maintain deadlock freedom
  - Turn model may not be feasible
    - Limited connectivity
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