Communication Performance in Network-on-Chips

Axel Jantsch
Royal Institute of Technology, Stockholm

August 2003
Overview

Introduction
Communication Performance
Organizational Structure
Interconnection Topologies
Trade-offs in Network Topology
Routing
Switch Design
Flow Control
NoC Examples
Introduction

- **Topology**: How switches and nodes are connected
- **Routing algorithm**: determines the route from source to destination
- **Switching strategy**: how a message traverses the route
- **Flow control**: Schedules the traversal of the message over time

A. Jantsch, KTH
Basic Definitions
Basic Definitions

**Message** is the basic communication entity.
Basic Definitions

**Message** is the basic communication entity.

**Flit** is the basic flow control unit. A message consists of 1 or many flits.
Basic Definitions

**Message** is the basic communication entity.

**Flit** is the basic flow control unit. A message consists of 1 or many flits.

**Phit** is the basic unit of the physical layer.
Basic Definitions

**Message** is the basic communication entity.

**Flit** is the basic flow control unit. A message consists of 1 or many flits.

**Phit** is the basic unit of the physical layer.

**Direct network** is a network where each switch connects to a node.
Basic Definitions

**Message** is the basic communication entity.

**Flit** is the basic flow control unit. A message consists of 1 or many flits.

**Phit** is the basic unit of the physical layer.

**Direct network** is a network where each switch connects to a node.

**Indirect network** is a network with switches not connected to any node.
Basic Definitions

**Message** is the basic communication entity.

**Flit** is the basic flow control unit. A message consists of 1 or many flits.

**Phit** is the basic unit of the physical layer.

**Direct network** is a network where each switch connects to a node.

**Indirect network** is a network with switches not connected to any node.

**Hop** is the basic communication action from node to switch or from switch to switch.
Basic Definitions

**Message** is the basic communication entity.
**Flit** is the basic flow control unit. A message consists of 1 or many flits.
**Phit** is the basic unit of the physical layer.
**Direct network** is a network where each switch connects to a node.
**Indirect network** is a network with switches not connected to any node.
**Hop** is the basic communication action from node to switch or from switch to switch.
**Diameter** is the length of the maximum shortest path between any two nodes measured in hops.
Basic Definitions

**Message** is the basic communication entity.

**Flit** is the basic flow control unit. A message consists of 1 or many flits.

**Phit** is the basic unit of the physical layer.

**Direct network** is a network where each switch connects to a node.

**Indirect network** is a network with switches not connected to any node.

**Hop** is the basic communication action from node to switch or from switch to switch.

**Diameter** is the length of the maximum shortest path between any two nodes measured in hops.

**Routing distance** between two nodes is the number of hops on a route.
Basic Definitions

**Message** is the basic communication entity.

**Flit** is the basic flow control unit. A message consists of 1 or many flits.

**Phit** is the basic unit of the physical layer.

**Direct network** is a network where each switch connects to a node.

**Indirect network** is a network with switches not connected to any node.

**Hop** is the basic communication action from node to switch or from switch to switch.

**Diameter** is the length of the maximum shortest path between any two nodes measured in hops.

**Routing distance** between two nodes is the number of hops on a route.

**Average distance** is the average of the routing distance over all pairs of nodes.
Basic Switching Techniques

Circuit Switching A real or virtual circuit establishes a direct connection between source and destination.

Packet Switching Each packet of a message is routed independently. The destination address has to be provided with each packet.

Store and Forward Packet Switching The entire packet is stored and then forwarded at each switch.

Cut Through Packet Switching The flits of a packet are pipelined through the network. The packet is not completely buffered in each switch.

Virtual Cut Through Packet Switching The entire packet is stored in a switch only when the header flit is blocked due to congestion.

Wormhole Switching is cut through switching and all flits are blocked on the spot when the header flit is blocked.
Latency

\[ \text{Time}(n) = \text{Admission} + \text{ChannelOccupancy} + \text{RoutingDelay} + \text{ContentionDelay} \]

**Admission** is the time it takes to emit the message into the network.

**ChannelOccupancy** is the time a channel is occupied.

**RoutingDelay** is the delay for the route.

**ContentionDelay** is the delay of a message due to contention.
Channel Occupancy

\[ \text{ChannelOccupancy} = \frac{n + n_E}{b} \]

- \( n \) ... message size in bits
- \( n_E \) ... envelop size in bits
- \( b \) ... raw bandwidth of the channel
Routing Delay

Store and Forward:

\[ T_{sf}(n, h) = h\left(\frac{n}{b} + \Delta\right) \]

Circuit Switching:

\[ T_{cs}(n, h) = \frac{n}{b} + h\Delta \]

Store and Forward with fragmented packets:

\[ T_{cs}(n, h, n_p) = \frac{n-n_p}{b} + h\left(\frac{n_p}{b} + \Delta\right) \]

Cut Through:

\[ T_{ct}(n, h) = \frac{n}{b} + h\Delta \]

\( n \) ... message size in bits
\( n_p \) ... size of message fragments in bits
\( h \) ... number of hops
\( b \) ... raw bandwidth of the channel
\( \Delta \) ... switching delay per hop
Routing Delay: Store and Forward vs Cut Through

SF vs CT switching; $d=2$, $k=10$, $b=1$

SF vs CT switching; $d=2$, $k=10$, $b=32$

SF vs CT switching, $k=2$, $m=8$

SF vs CT switching, $d=2$, $m=8$

Cut Through
Store and Forward

A. Jantsch, KTH
Local and Global Bandwidth

Local bandwidth \[ = b \left( \frac{n}{n+n_E+w\Delta} \right) \]

Total bandwidth \[ = Cb [\text{bits/second}] = Cw [\text{bits/cycle}] = C [\text{phits/cycle}] \]

Bisection bandwidth

... minimum bandwidth to cut the net into two equal parts.

- \( b \) ... raw bandwidth of a link;
- \( n \) ... message size;
- \( n_E \) ... size of message envelope;
- \( w \) ... link bandwidth per cycle;
- \( \Delta \) ... switching time for each switch in cycles;
- \( w\Delta \) ... bandwidth lost during switching;
- \( C \) ... total number of channels;

For a \( k \times k \) mesh with bidirectional channels:

Total bandwidth \[ = (4k^2 - 4k)b \]

Bisection bandwidth \[ = 2kb \]
Link and Network Utilization

**total load on the network:** \[ L = \frac{Nhl}{M} \text{[phits/cycle]} \]

**load per channel:** \[ \rho = \frac{Nhl}{MC} \text{[phits/cycle]} \leq 1 \]

- \( M \) … each host issues a packet every \( M \) cycles
- \( C \) … number of channels
- \( N \) … number of nodes
- \( h \) … average routing distance
- \( l = n/w \) … number of cycles a message occupies a channel
- \( n \) … average message size
- \( w \) … bitwidth per channel
Network Saturation

Typical saturation points are between 40% and 70%. The saturation point depends on

- Traffic pattern
- Stochastic variations in traffic
- Routing algorithm
Organizational Structure

- Link
- Switch
- Network Interface
Link

**Short link** At any time there is only one data word on the link.

**Long link** Several data words can travel on the link simultaneously.

**Narrow link** Data and control information is multiplexed on the same wires.

**Wide link** Data and control information is transmitted in parallel and simultaneously.

**Synchronous clocking** Both source and destination operate on the same clock.

**Asynchronous clocking** The clock is encoded in the transmitted data to allow the receiver to sample at the right time instance.
Switch

Input ports

Receiver
Input buffer

Crossbar

Output buffer

Transmitter
Output ports

Control
(Routing, Scheduling)
Switch Design Issues

Degree: number of inputs and outputs;

Buffering
- Input buffers
- Output buffers
- Shared buffers

Routing
- Source routing
- Deterministic routing
- Adaptive routing

Output scheduling

Deadlock handling

Control flow

A. Jantsch, KTH
Network Interface

- Admission protocol
- Reception obligations
- Buffering
- Assembling and disassembling of messages
- Routing
- Higher level services and protocols
Interconnection Topologies

- Fully connected networks
- Linear arrays and rings
- Multidimensional meshes and tori
- Trees
- Butterflies
**Fully Connected Networks**

**Bus:**
- Switch degree: $N$
- Diameter: 1
- Distance: 1
- Network cost: $O(N)$
- Total bandwidth: $b$
- Bisection bandwidth: $b$

**Crossbar:**
- Switch degree: $N$
- Diameter: 1
- Distance: 1
- Network cost: $O(N^2)$
- Total bandwidth: $Nb$
- Bisection bandwidth: $Nb$
Linear Arrays and Rings

Linear array:
- switch degree = 2
- diameter = \( N - 1 \)
- distance \( \approx 2/3N \)
- network cost = \( O(N) \)
- total bandwidth = \( 2(N - 1)b \)
- bisection bandwidth

Torus:
- switch degree = 2
- diameter = \( N/2 \)
- distance \( \approx 1/3N \)
- network cost = \( O(N) \)
- total bandwidth = \( 2N b \)
- bisection bandwidth = \( 4b \)

A. Jantsch, KTH
Multidimensional Meshes and Tori

\textit{k-ary d-cubes} are \(d\)-dimensional tori with unidirectional links and \(k\) nodes in each dimension:

- number of nodes \(N = k^d\)
- switch degree \(= d\)
- diameter \(= d(k - 1)\)
- distance \(\sim d \frac{1}{2} (k - 1)\)
- network cost \(= O(N)\)
- total bandwidth \(= 2Nb\)
- bisection bandwidth \(= 2k^{(d-1)}b\)

A. Jantsch, KTH
Routing Distance in $k$-ary $n$-Cubes

Network Scalability wrt Distance

A. Jantsch, KTH
Projecting High Dimensional Cubes

2-ary 2-cube

2-ary 3-cube

2-ary 4-cube

2-ary 5-cube
Binary Trees

- Number of nodes $N = 2^d$
- Number of switches $= 2^d - 1$
- Switch degree $= 3$
- Diameter $= 2d$
- Distance $\sim d + 2$
- Network cost $= O(N)$
- Total bandwidth $= 2 \cdot 2(N - 1)b$
- Bisection bandwidth $= 2b$
$k$-ary Trees

- Number of nodes $N = k^d$
- Number of switches $\sim k^d$
- Switch degree $= k + 1$
- Diameter $= 2d$
- Distance $\sim d + 2$
- Network cost $= O(N)$
- Total bandwidth $= 2 \cdot 2(N - 1)b$
- Bisection bandwidth $= kb$
Binary Tree Projection

- Efficient and regular 2-layout;
- Longest wires in resource width:

\[ lW = 2^{\frac{d-1}{2}} - 1 \]

<table>
<thead>
<tr>
<th>$d$</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td>128</td>
<td>256</td>
<td>512</td>
<td>1024</td>
</tr>
<tr>
<td>$lW$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
### k-ary n-Cubes versus k-ary Trees

#### k-ary n-cubes:

<table>
<thead>
<tr>
<th>Property</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of nodes</td>
<td>$N = k^d$</td>
</tr>
<tr>
<td>Switch degree</td>
<td>$d + 2$</td>
</tr>
<tr>
<td>Diameter</td>
<td>$d(k - 1)$</td>
</tr>
<tr>
<td>Distance</td>
<td>$d^{1/2}(k - 1)$</td>
</tr>
<tr>
<td>Network cost</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>Total bandwidth</td>
<td>$2N b$</td>
</tr>
<tr>
<td>Bisection bandwidth</td>
<td>$2k^{(d-1)} b$</td>
</tr>
</tbody>
</table>

#### k-ary trees:

<table>
<thead>
<tr>
<th>Property</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of nodes</td>
<td>$N = k^d$</td>
</tr>
<tr>
<td>Number of switches</td>
<td>$\sim k^d$</td>
</tr>
<tr>
<td>Switch degree</td>
<td>$k + 1$</td>
</tr>
<tr>
<td>Diameter</td>
<td>$2d$</td>
</tr>
<tr>
<td>Distance</td>
<td>$d + 2$</td>
</tr>
<tr>
<td>Network cost</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>Total bandwidth</td>
<td>$2 \cdot 2(N - 1)b$</td>
</tr>
<tr>
<td>Bisection bandwidth</td>
<td>$kb$</td>
</tr>
</tbody>
</table>
Butterflies

Butterfly building block

16 node butterfly
**Butterfly Characteristics**

- **number of nodes** $N = 2^d$
- **number of switches** = $2^{d-1}d$
- **switch degree** = 2
- **diameter** = $d + 1$
- **distance** = $d + 1$
- **network cost** = $O(Nd)$
- **total bandwidth** = $2^d db$
- **bisection bandwidth** = $\frac{N}{2} b$
### $k$-ary $n$-Cubes versus $k$-ary Trees vs Butterflies

<table>
<thead>
<tr>
<th></th>
<th>$k$-ary $n$-cubes</th>
<th>binary tree</th>
<th>butterfly</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost per node</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
<td>$O(N \log N)$</td>
</tr>
<tr>
<td>distance</td>
<td>$\frac{d}{2} \sqrt{N} \log N$</td>
<td>$2 \log N$</td>
<td>$\log N$</td>
</tr>
<tr>
<td>links per node</td>
<td>2</td>
<td>2</td>
<td>$\log N$</td>
</tr>
<tr>
<td>bisection</td>
<td>$2N \frac{d-1}{d}$</td>
<td>1</td>
<td>$\frac{1}{2}N$</td>
</tr>
<tr>
<td>frequency limit of random traffic</td>
<td>$\sqrt{\frac{N}{2}}$</td>
<td>$N$</td>
<td>2</td>
</tr>
</tbody>
</table>

A. Jantsch, KTH
Problems with Butterflies

• Cost of the network
  ★ $O(N \log N)$
  ★ 2-d layout is more difficult than for binary trees
  ★ Number of long wires grows faster than for trees.

• For each source-destination pair there is only one route.
• Each route blocks many other routes.
Benes Networks

- Many routes;
- Costly to compute non-blocking routes;
- High probability for non-blocking route by randomly selecting an intermediate node [Leighton, 1992];
Fat Trees

16-node 2-ary fat-tree
**k-ary n-dimensional Fat Tree Characteristics**

- number of nodes $N = k^d$
- number of switches $= k^{d-1}d$
- switch degree $= 2k$
- diameter $= 2d$
- distance $\approx d$
- network cost $= O(Nd)$
- total bandwidth $= 2k^d db$
- bisection bandwidth $= 2k^{d-1}b$

16-node 2-ary fat-tree

A. Jantsch, KTH
**k-ary n-Cubes versus k-ary d-dimensional Fat Trees**

**k-ary n-cubes:**
- Number of nodes \( N = k^d \)
- Switch degree \( = d \)
- Diameter \( = d(k - 1) \)
- Distance \( \sim d^{1/2}(k - 1) \)
- Network cost \( = O(N) \)
- Total bandwidth \( = 2Nb \)
- Bisection bandwidth \( = 2k^{(d-1)}b \)

**k-ary n-dimensional fat trees:**
- Number of nodes \( N = k^d \)
- Number of switches \( = k^{d-1}d \)
- Switch degree \( = 2k \)
- Diameter \( = 2d \)
- Distance \( \sim d \)
- Network cost \( = O(Nd) \)
- Total bandwidth \( = 2k^d db \)
- Bisection bandwidth \( = 2k^{d-1}b \)
Relation between Fat Tree and Hypercube

binary 2–dim fat tree

binary 1–cube

A. Jantsch, KTH
Relation between Fat Tree and Hypercube - cont’d

binary 3–dim fat tree

binary 2–cube

binary 2–cube
Relation between Fat Tree and Hypercube - cont’d

binary 4-dim fat tree

binary 3-cube

binary 3-cube
## Topologies of Parallel Computers

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>nCUBE/2</td>
<td>Hypercube</td>
<td>25</td>
<td>1</td>
<td>40</td>
<td>32</td>
</tr>
<tr>
<td>TMC CM-5</td>
<td>Fat tree</td>
<td>25</td>
<td>4</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>IBM SP-2</td>
<td>Banyan</td>
<td>25</td>
<td>8</td>
<td>5</td>
<td>16</td>
</tr>
<tr>
<td>Intel Paragon</td>
<td>2D Mesh</td>
<td>11.5</td>
<td>16</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>Meiko CS-2</td>
<td>Fat tree</td>
<td>20</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Cray T3D</td>
<td>3D Torus</td>
<td>6.67</td>
<td>16</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>DASH</td>
<td>Torus</td>
<td>30</td>
<td>16</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>J-Machine</td>
<td>3D Mesh</td>
<td>31</td>
<td>8</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>Monsoon</td>
<td>Butterfly</td>
<td>20</td>
<td>16</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>SGI Origin</td>
<td>Hypercube</td>
<td>2.5</td>
<td>20</td>
<td>16</td>
<td>160</td>
</tr>
<tr>
<td>Myricom</td>
<td>Arbitrary</td>
<td>6.25</td>
<td>16</td>
<td>50</td>
<td>16</td>
</tr>
</tbody>
</table>

A. Jantsch, KTH
Trade-offs in Topology Design for the $k$-ary $n$-Cube

- Unloaded Latency
- Latency under Load
Network Scaling for Unloaded Latency

Latency\( (n) \) = Admission + ChannelOccupancy + RoutingDelay + ContentionDelay

RoutingDelay \( T_{ct}(n, h) \) = \( \frac{n}{b} + h\Delta \)

RoutingDistance \( h \) = \( \frac{1}{2}d(k - 1) = \frac{1}{2}(k - 1) \log_k N = \frac{1}{2}(d\sqrt{N} - 1) \)
Unloaded Latency for Small Networks and Local Traffic

Network scalability wrt latency (m=128)

Number of nodes

Average latency

Network scalability wrt latency (m=128; h=dk/5)

Number of nodes

Average latency
Unloaded Latency under a Free-Wire Cost Model

**Free-wire** cost model: Wires are free and can be added without penalty.

---

A. Jantsch, KTH
Unloaded Latency under a Fixed-Wire Cost Models

**Fixed-wire** cost model: The number of wires is constant per node:
128 wires per node: \( w(d) = \left\lfloor \frac{64}{d} \right\rfloor \).

\[
\begin{array}{cccccccccc}
  d & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
  w(d) & 32 & 21 & 16 & 12 & 10 & 9 & 8 & 7 & 6 \\
\end{array}
\]
Unloaded Latency under a Fixed-Bisection Cost Models

**Fixed-bisection** cost model: The number of wires across the bisection is constant:

\[ \text{bisection} = 1024 \text{ wires: } w(d) = \frac{k}{2} = \frac{d\sqrt{N}}{2}. \]

Example: \( N=1024 \):

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c}
    d & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
    w(d) & 512 & 16 & 5 & 3 & 2 & 2 & 1 & 1 & 1 \\
\end{array}
\]
Unloaded Latency under a Logarithmic Wire Delay Cost Models

Fixed-bisection Logarithmic Wire Delay cost model: The number of wires across the bisection is constant and the delay on wires increases logarithmically with the length [Dally, 1990]:

Length of long wires: \( l = k^{\frac{n}{2}} - 1 \)

\[
T_c \propto 1 + \log l = 1 + \left( \frac{d}{2} - 1 \right) \log k
\]
Unloaded Latency under a Linear Wire Delay Cost Models

**Fixed-bisection Linear Wire Delay** cost model: The number of wires across the bisection is constant and the delay on wires increases linearly with the length [Dally, 1990]:

Length of long wires: \( l = k^{\frac{n}{2}} - 1 \)

\[ T_c \propto l = k^{\frac{d}{2}} - 1 \]
Latency under Load

Assumptions [Agarwal, 1991]:

- $k$-ary $n$-cubes
- random traffic
- dimension-order cut-through routing
- unbounded internal buffers (to ignore flow control and deadlock issues)
Latency under Load - cont’d

Latency(n) = Admission + ChannelOccupancy + RoutingDelay + ContentionDelay

\[ T(m, k, d, w, \rho) = \text{RoutingDelay} + \text{ContentionDelay} \]

\[ T(m, k, d, w, \rho) = \frac{m}{w} + dh_k(\Delta + W(m, k, d, w, \rho)) \]

\[ W(m, k, d, w, \rho) = \frac{m}{w} \cdot \frac{\rho}{1 - \rho} \cdot \frac{h_k - 1}{h_k^2} \cdot \left(1 + \frac{1}{d}\right) \]

\[ h = \frac{1}{2}d(k - 1) \]

\[ m \cdots \text{message size} \]
\[ w \cdots \text{bitwidth of link} \]
\[ \rho \cdots \text{aggregate channel utilization} \]
\[ h_k \cdots \text{average distance in each dimension} \]
\[ \Delta \cdots \text{switching time in cycles} \]
Routing

**Deterministic routing** The route is determined solely by source and destination locations.

**Arithmetic routing** The destination address of the incoming packet is compared with the address of the switch and the packet is routed accordingly. (relative or absolute addresses)

**Source based routing** The source determines the route and builds a header with one directive for each switch. The switches strip off the top directive.

**Table-driven routing** Switches have routing tables, which can be configured.

**Adaptive routing** The route can be adapted by the switches to balance the load.

**Minimal routing** allows only shortest paths while non-minimal routing allows even longer paths.
Deadlock

**Deadlock** Two or several packets mutually block each other and wait for resources, which can never be free.

**Livelock** A packet keeps moving through the network but never reaches its destination.

**Starvation** A packet never gets a resource because it always looses the competition for that resource (fairness).
Deadlock Situations

- Head-on deadlock;
- Nodes stop receiving packets;
- Contention for switch buffers can occur with store-and-forward, virtual-cut-through and wormhole routing. Wormhole routing is particularly sensible.
- Cannot occur in butterflies;
- Cannot occur in trees or fat trees if upward and downward channels are independent;
- Dimension order routing is deadlock free on $k$-ary $n$-arrays but not on tori with any $n \geq 1$. 
Deadlock in a 1-dimensional Torus

Message 1 from C→ B, 10 flits
Message 2 from A→ D, 10 flits
Routing is deadlock free if the channel dependence graph has no cycles.
Deadlock-free Routing

- Two main approaches:
  - Restrict the legal routes;
  - Restrict how resources are allocated;
- Number the channel cleverly
- Construct the channel dependence graph
- Prove that all legal routes follow a strictly increasing path in the channel dependence graph.
Virtual Channels

- Virtual channels can be used to break cycles in the dependence graph.
- E.g. all $n$-dimensional tori can be made deadlock free under dimension-order routing by assigning all wrap-around paths to a different virtual channel than other links.
Virtual Channels and Deadlocks
Turn-Model Routing

What are the minimal routing restrictions to make routing deadlock free?

- Three minimal routing restriction schemes:
  - North-last
  - West-first
  - Negative-first

- Allow complex, non-minimal adaptive routes.
- Unidirectional \( k \)-ary \( n \)-cubes still need virtual channels.
Adaptive Routing

- The switch makes routing decisions based on the load.
- Fully adaptive routing allows all shortest paths.
- Partial adaptive routing allows only a subset of the shortest path.
- Non-minimal adaptive routing allows also non-minimal paths.
- Hot-potato routing is non-minimal adaptive routing without packet buffering.
Switch Design
Switch Design
Switch Design

Control lines
Switch Design

I\textsubscript{0} \rightarrow O\textsubscript{0}

I\textsubscript{1} \rightarrow O\textsubscript{1}

I\textsubscript{2} \rightarrow O\textsubscript{2}

I\textsubscript{3} \rightarrow O\textsubscript{3}

Control lines

Address \rightarrow \text{RAM}

\begin{align*}
I\textsubscript{0} & \rightarrow O\textsubscript{0} \\
I\textsubscript{1} & \rightarrow O\textsubscript{1} \\
I\textsubscript{2} & \rightarrow O\textsubscript{2} \\
I\textsubscript{3} & \rightarrow O\textsubscript{3}
\end{align*}
Switch Scaling

- Switches are *wire dominated*
Switch Scaling

- Switches are **wire dominated**
- Scaling with equal switch size by a factor \( s \):
  - number of transistors: \( s^2 \)
  - number of I/O's: \( s \)
  - speed of transistor: \( 1/s \)
  - speed of wires: 1
  - **delay of the switch:** 1
Switch Scaling

- Switches are **wire dominated**
- Scaling with equal switch size by a factor \( s \):
  - number of transistors: \( s^2 \)
  - number of I/O's: \( s \)
  - speed of transistor: \( 1/s \)
  - speed of wires: 1
  - **delay of the switch**: 1
- Scaling with equal I/O number by factor \( s \):
  - size of the switch: \( 1/s^2 \)
  - speed of transistor: \( 1/s \)
  - speed of wires: \( 1/s \)
  - **delay of the switch**: \( 1/s \)
Buffering

- Input buffering
- Output buffering
- Shared buffers
- Virtual channel buffers
**Input Buffering**

Head-of line blocking limits output channel utilization to 60%.
Output Buffering

A. Jantsch, KTH
Shared Buffer Pool

- Potential better utilization of buffers;
- Speed of memory becomes limiting factor;
Virtual Channel Buffering

- Dynamic virtual channel allocation can
  - increase buffer utilization,
  - reduce head-of-line blocking.
- E.g. 256 nodes 2-ary butterfly with wormhole routing; 16 flits per link:
  - no virtual channel: output channel saturation at 25% under random traffic;
  - 16 virtual channels: saturation at 80% traffic load;

A. Jantsch, KTH
Output arbitration policy options:

**Static priority**  Simple implementation; Potential for starvation;
**Round-robin**   requires additional state;
**Random**
**Oldest-first**  Same average latency as Random, but less variation;
Output Scheduling and Routing Algorithms

- Restricted routing directions avoid full connectivity;
- Adaptive routing allows an input to request to several/all outputs
Stacked Dimension Switch

- Simple $2 \times 2$ building block can be repeatedly used;
- in $k$-ary $n$-cubes we need $n$ $2 \times 2$ switches instead of $n \times n$ switches;
- Changing dimension incurs additional delay;
- Switching is very fast in the same dimension;
Flow Control

When a packet contends for a shared resource (link, buffer) we have three principle options:

- Buffering the packet and stalling new traffic;
- Dropping the packet;
- Allocating an alternative resource;
Flow Control

When a packet contends for a shared resource (link, buffer) we have three principle options:

- Buffering the packet and stalling new traffic;
- Dropping the packet;
- Allocating an alternative resource;

- Link level flow control;
- End to end flow controls;
Link Level Flow Control

Short, wide link

Long, wide link

Long, narrow link

A. Jantsch, KTH
Flow Control with Watermarks
End-to-end Flow Control

- Link level flow control can manage short term imbalances.
- Long term imbalances (more data is injected than drained) must be addressed with end-to-end flow control.
Source-Destination Inbalance

Deterministic Routing
Source-Destination Inbalance

Deterministic Routing

Minimal Adaptive Routing

A. Jantsch, KTH
Source-Destination Inbalance

- Deterministic Routing
- Minimal Adaptive Routing
- Nonminimal Adaptive Routing
Source-Destination Inbalance

Congestion causes:

- Source-destination imbalance;
- Hot spots;
- Random overload;
End-to-end Protocols and Admission Control

- Acknowledgement based protocols;
- Credit based protocols;
- Threshold based network admission protocols;
Summary

- Communication Performance: bandwidth, unloaded latency, loaded latency
- Organizational Structure: NI, switch, link
- Topologies: wire space and delay domination favors low dimension topologies;
- Routing: deterministic vs source based vs adaptive routing; deadlock;
- Switch: Buffering; output scheduling; flow control;
- Flow control: Link level and end-to-end control;
Issues beyond the Scope of this Lecture

- Power
- Clocking
- Faults and reliability
- Memory architecture and I/O
- Application specific communication patterns
- Services offered to applications; Quality of service
NoC Research Projects

- Nostrum at KTH
- Æthereal at Philips Research
- Proteo at Tampere University of Technology
- SPIN at UPMC/LIP6 in Paris
- XPipes at Bologna U
- Octagon at ST and UC San Diego
Nostrum (KTH)

- 2-d mesh topology
- Wide (128 bits), short links
- Non-minimal adaptive hot-potato routing
- No buffering
- Services: Best effort, guaranteed latency virtual circuits
- Four data protection levels at link layer

(from [Nilsson et al., 2003])
Æthereal (Philips)

- Topology: probably low dimensional tree or mesh, or dedicated
- Deterministic source based routing
- Wormhole or virtual cut-through switching
- Input buffering
- Selectable connection features:
  - Integrity
  - Completion
  - Ordering
  - Bounds on latency, throughput and jitter

(from [Goossens et al., 2003])
SPIN (UPMC/LIP6)

- Topology: Fat tree
- Wormhole switching
- Adaptive routing
- Input buffering
- Bidirectional 32 bit links
- Separate control network for configuration

[Adriahantenaina et al., 2003]
Octagon (UCD, ST)

- Basic 8-node architecture
- Diameter: 2 hops
- Packet and circuit switching modes
- Source based routing with a 3 bit address in the header

from [Karim et al., 2001]
X Pipes (Bologna U)

- Source based routing
- Wormhole switching
- Long, pipelined links
- Output buffering
- Virtual channels
- Acknowledgment based flow control with retransmission
• Objective is to develop a flexible library of communication IPs that support various topologies and routing strategies.

• Topology: Ring connecting local bus or star based clusters;

• Virtual cut-through switching

• Routing: Table based

• Buffering: Input and output buffering

(from [Alho and Nurmi, 2003])
To Probe Further - Books and Classic Papers


To Probe Further - NoC Examples


