On-Chip Networks I: Topology/Flow Control

Daniel Sanchez
Computer Science & Artificial Intelligence Lab
M.I.T.
The Shift to Multicore

[Produced with CPUDB, cpubdb.stanford.edu]
The Shift to Multicore

- Since 2005, improvements in system performance mainly due to increasing cores/chip
The Shift to Multicore

- Since 2005, improvements in system performance mainly due to increasing cores/chip
- Why?
The Shift to Multicore

- Since 2005, improvements in system performance mainly due to increasing cores/chip
- Why? Limited instruction-level parallelism
The Shift to Multicore

- Since 2005, improvements in system performance mainly due to increasing cores/chip
- Why? Limited instruction-level parallelism
  Technology scaling
Classic CMOS Scaling

- Moore’s law + Dennard scaling: Each generation (e.g., 90→65nm),

\[ P = P_{\text{dyn}} + P_{\text{leak}} \]

\[ P_{\text{dyn}} = \frac{1}{2} CV^2 f \]

\[ P_{\text{leak}} = V \times I_{\text{leak}} \ll P_{\text{dyn}} \]

[Adapted from “Advancing Systems Without Technology Progress” outbrief of DARPA/ISAT Workshop, 2012]
Classic CMOS Scaling

- Moore’s law + Dennard scaling: Each generation (e.g., 90→65nm),

\[ P = P_{\text{dyn}} + P_{\text{leak}} \]

\[ P_{\text{dyn}} = \frac{1}{2} CV^2 f \]

\[ P_{\text{leak}} = V \times I_{\text{leak}} \ll P_{\text{dyn}} \]

[Adapted from “Advancing Systems Without Technology Progress” outbrief of DARPA/ISAT Workshop, 2012]
Classic CMOS Scaling

- Moore’s law + Dennard scaling: Each generation (e.g., 90→65nm),

\[ P = P_{\text{dyn}} + P_{\text{leak}} \]

\[ P_{\text{dyn}} = \frac{1}{2} CV^2 f \]

\[ P_{\text{leak}} = V \times I_{\text{leak}} \ll P_{\text{dyn}} \]

[Adapted from “Advancing Systems Without Technology Progress” outbrief of DARPA/ISAT Workshop, 2012]
Classic CMOS Scaling

- Moore’s law + Dennard scaling: Each generation (e.g., 90→65nm),

\[
P = P_{\text{dyn}} + P_{\text{leak}}
\]

\[
P_{\text{dyn}} = \frac{1}{2} CV^2 f
\]

\[
P_{\text{leak}} = V \times I_{\text{leak}} \ll P_{\text{dyn}}
\]

[Adapted from “Advancing Systems Without Technology Progress” outbrief of DARPA/ISAT Workshop, 2012]
Classic CMOS Scaling

- Moore’s law + Dennard scaling: Each generation (e.g., 90\(\rightarrow\)65nm),

\[
P = P_{\text{dyn}} + P_{\text{leak}}
\]

\[
P_{\text{dyn}} = \frac{1}{2}CV^2f
\]

\[
P_{\text{leak}} = V \times I_{\text{leak}} \ll P_{\text{dyn}}
\]

2x transistors, 1.4x frequency, same power \(\rightarrow\) area-constrained designs

[Adapted from “Advancing Systems Without Technology Progress” outbrief of DARPA/ISAT Workshop, 2012]
Current CMOS Scaling

- Frequency and supply voltage scaling are mostly exhausted

2x transistors, same frequency, 1.4x power → power-constrained designs

[Adapted from “Advancing Systems Without Technology Progress” outbrief of DARPA/ISAT Workshop, 2012]
Multicore Performance

![Cost/perf curve of possible core designs](image)

- High-perf, expensive core

Cost (area, energy...)

Performance
Multicore Performance

Cost (area, energy...)

High-perf, expensive core

Cost/perf curve of possible core designs

Moderate perf, efficient core

Performance
Multicore Performance

Cost (area, energy...) vs Performance

- High-perf, expensive core
- Moderate perf, efficient core
- Cost/perf curve of possible core designs
- 2 cores
Multicore Performance

Cost (area, energy...) vs. Performance

- High-perf, expensive core
- Cost/perf curve of possible core designs
- Moderate perf, efficient core
- 2 cores
- 4 cores

April 1, 2015
What factors may limit multicore performance?
Multicore Performance

Cost (area, energy...)

Performance

Cost/perf curve of possible core designs

High-perf, expensive core

Moderate perf, efficient core

2 cores

4 cores

What factors may limit multicore performance?

Limited application parallelism

April 1, 2015
Multicore Performance

What factors may limit multicore performance?

- Limited application parallelism
- Memory accesses and inter-core communication
Multicore Performance

What factors may limit multicore performance?

- Limited application parallelism
- Memory accesses and inter-core communication
- Programming complexity
Amdahl’s Law

- Speedup = \( \frac{\text{time}_{\text{without enhancement}}}{\text{time}_{\text{with enhancement}}} \)
- Suppose an enhancement speeds up a fraction \( f \) of a task by a factor of \( S \)

\[
\text{time}_{\text{new}} = \text{time}_{\text{old}} \cdot \left( (1-f) + \frac{f}{S} \right)
\]

\[
S_{\text{overall}} = \frac{1}{(1-f) + \frac{f}{S}}
\]
Amdahl’s Law

- Speedup = \( \frac{\text{time}_{\text{without enhancement}}}{\text{time}_{\text{with enhancement}}} \)
- Suppose an enhancement speeds up a fraction \( f \) of a task by a factor of \( S \)
  \[
  \text{time}_{\text{new}} = \text{time}_{\text{old}} \cdot \left( (1-f) + \frac{f}{S} \right) 
  
  S_{\text{overall}} = \frac{1}{(1-f) + \frac{f}{S}}
  
  \]

Corollary: Make the common case fast
Amdahl’s Law and Parallelism

• Say you write a program that can do 90% of the work in parallel, but the other 10% is sequential.

• What is the maximum speedup you can get by running on a multicore machine?
Amdahl’s Law and Parallelism

- Say you write a program that can do 90% of the work in parallel, but the other 10% is sequential.
- What is the maximum speedup you can get by running on a multicore machine?

\[ S_{\text{overall}} = \frac{1}{(1-f) + \frac{f}{S}} \]
Amdahl’s Law and Parallelism

• Say you write a program that can do 90% of the work in parallel, but the other 10% is sequential.

• What is the maximum speedup you can get by running on a multicore machine?

\[ S_{\text{overall}} = \frac{1}{(1-f) + \frac{f}{S}} \]

\[ f = 0.9, \ S=\infty \rightarrow S_{\text{overall}} = 10 \]
Amdahl’s Law and Parallelism

• Say you write a program that can do 90% of the work in parallel, but the other 10% is sequential

• What is the maximum speedup you can get by running on a multicore machine?

\[
S_{\text{overall}} = \frac{1}{(1-f) + \frac{f}{S}}
\]

\[f = 0.9, \ S = \infty \Rightarrow S_{\text{overall}} = 10\]

What \( f \) do you need to use a 1000-core machine well?
On-Chip Networks
History: From interconnection networks to on-chip networks

Box-to-box networks

Board-to-board networks

Chip-to-chip networks

On-chip networks

Multi-Chip: Supercomputers, Data Centers, Internet Routers, Servers

On-Chip: Servers, Laptops, Phones, HDTVs, Access routers

April 1, 2015
History: From interconnection networks to on-chip networks

Box-to-box networks

Board-to-board networks

Chip-to-chip networks

On-chip networks

Focus on on-chip networks connecting caches in shared memory processors

Multi-Chip: Supercomputers, Data Centers, Internet Routers, Servers
On-Chip: Servers, Laptops, Phones, HDTVs, Access routers
What’s an on-chip network?

E.g. Cache-coherent chip multiprocessor

Load reg1, addressA

On-Chip Network

Home node for address A

Sharer that holds a copy of address A in its $
What’s an on-chip network?

E.g. Cache-coherent chip multiprocessor

Load \texttt{reg1, addressA}

It transports cache coherence messages and cache lines between processor cores

Home node for \texttt{address A}

Sharer that holds a copy of \texttt{address A in its $}
Designing an on-chip network
Designing an on-chip network

• Topology
Designing an on-chip network

• Topology
Designing an on-chip network

- Topology
Designing an on-chip network

• Topology
Designing an on-chip network

- Topology
- Routing
Designing an on-chip network

- Topology

- Routing
Designing an on-chip network

- Topology
- Routing
Designing an on-chip network

- Topology
- Flow control
- Routing
Designing an on-chip network

- Topology
- Flow control
- Routing
Designing an on-chip network

- Topology
- Flow control
- Routing
Designing an on-chip network

- Topology
- Flow control
- Router microarchitecture
- Routing
Designing an on-chip network

- Topology
- Flow control
- Router microarchitecture
- Routing
Interconnection Network Architecture

• **Topology**: How to connect the nodes up? (processors, memories, router line cards, ...)
• **Routing**: Which path should a message take?
• **Flow control**: How is the message actually forwarded from source to destination?
• **Router microarchitecture**: How to build the routers?
• **Link microarchitecture**: How to build the links?
Topology
Topological Properties

- *Routing Distance* - number of links on route
- *Diameter* - maximum routing distance
- *Average Distance*
- A network is *partitioned* by a set of links if their removal disconnects the graph
- *Bisection Bandwidth* is the bandwidth crossing a minimal cut that divides the network in half
Linear Arrays and Rings

Route A -> B given by relative address $R = B - A$

Diameter?
Average distance?
Bisection bandwidth?

• Torus Examples:
  - FDDI, SCI, FiberChannel Arbitrated Loop, Intel Xeon
Linear Arrays and Rings

Route A -> B given by relative address $R = B - A$

- **Linear Array**
- **Torus**
- **Torus arranged to use short wires**

- Diameter?
- Average distance?
- Bisection bandwidth?

- **Torus Examples:**
  - FDDI, SCI, FiberChannel Arbitrated Loop, Intel Xeon
Linear Arrays and Rings

Route A -> B given by relative address $R = B - A$

- Diameter?
- Average distance?
- Bisection bandwidth?

- Torus Examples:
  - FDDI, SCI, FiberChannel Arbitrated Loop, Intel Xeon
Linear Arrays and Rings

Route A -> B given by relative address $R = B - A$

Linear Array  Ring (1-D Torus)

Diameter?  $N-1$
Average distance?  $N/3 - 1/(3N)$
Bisection bandwidth?  1

• Torus Examples:
  - FDDI, SCI, FiberChannel Arbitrated Loop, Intel Xeon
Linear Arrays and Rings

Route A -> B given by relative address \( R = B - A \)

Linear Array

Torus

Torus arranged to use short wires

- Diameter?
- Average distance?
- Bisection bandwidth?

- Linear Array: \( N-1 \)
- Ring (1-D Torus): \( N/2 \) (if even \( N \))
- \( N/3-1/(3N) \)
- 1

- Torus Examples:
  - FDDI, SCI, FiberChannel Arbitrated Loop, Intel Xeon
Route A -> B given by relative address \( R = B - A \)

- **Linear Arrays**
  - Diameter? \( N-1 \)
  - Average distance? \( \frac{N-1}{3N} \)
  - Bisection bandwidth? \( 1 \)

- **Rings (1-D Torus)**
  - Diameter? \( \frac{N}{2} \) (if even \( N \))
  - Average distance? \( \frac{N-1}{3N} \)
  - Bisection bandwidth? \( \frac{N}{4} \) (if even \( N \))

- **Torus Examples:**
  - FDDI, SCI, FiberChannel Arbitrated Loop, Intel Xeon

- **Torus**

- **Torus arranged to use short wires**

---

April 1, 2015
Linear Arrays and Rings

Route A -> B given by relative address $R = B - A$

Linear Array

Torus

Torus arranged to use short wires

Linear Array    Ring (1-D Torus)

Diameter?    N-1
Average distance?    $N/3 - 1/(3N)$
Bisection bandwidth?    1

N/2  (if even N)
N/4  (if even N)
2

• Torus Examples:
  - FDDI, SCI, FiberChannel Arbitrated Loop, Intel Xeon
Multidimensional Meshes and Tori

- **d-dimensional array**
  - \( n = k_{d-1} \times \ldots \times k_0 \) nodes
  - described by \( d \)-vector of coordinates \((i_{d-1}, \ldots, i_0)\)

- **d-dimensional \( k \)-ary mesh**: \( N = k^d \)
  - \( k = \sqrt[d]{N} \)
  - described by \( d \)-vector of radix \( k \) coordinate

- **d-dimensional \( k \)-ary torus** (or \( k \)-ary \( d \)-cube)
Routing & Flow Control Overview
Messages, Packets, Flits, Phits

Packet: Basic unit of routing and sequencing
- Limited size (e.g. 64 bits – 64 KB)
Flit (flow control digit): Basic unit of bandwidth/storage allocation
- All flits in packet follow the same path
Phit (physical transfer digit): data transferred in single clock

Why flits?

Packet: Basic unit of routing and sequencing
- Limited size (e.g. 64 bits – 64 KB)
Flit (flow control digit): Basic unit of bandwidth/storage allocation
- All flits in packet follow the same path
Phit (physical transfer digit): data transferred in single clock
Messages, Packets, Flits, Phits

Packet: Basic unit of routing and sequencing
- Limited size (e.g. 64 bits – 64 KB)

Flit (flow control digit): Basic unit of bandwidth/storage allocation
- All flits in packet follow the same path

Phit (physical transfer digit): data transferred in single clock

Why flits?

For variable packet sizes
Routing vs Flow Control

- Routing algorithm chooses path that packets should follow to get from source to destination

- Flow control schemes allocate resources (buffers, links, control state) to packets traversing the network

- Our approach: Bottom-up
  - Today: Flow control, assuming routes are set
  - Next lecture: Routing algorithms
Properties of Routing Algorithms

- **Deterministic/Oblivious**
  - Route determined by (source, dest), not intermediate state (i.e. traffic)

- **Adaptive**
  - Route influenced by traffic along the way

- **Minimal**
  - Only selects shortest paths

- **Deadlock-free**
  - No traffic pattern can lead to a situation where no packets move forward

(more in next lecture)
Flow Control
Contention

- Two packets trying to use the same link at the same time
  - Limited or no buffering
- Problem arises because we are sharing resources
  - Sharing bandwidth and buffering
Flow Control Protocols

- **Bufferless**
  - Circuit switching
  - Dropping
  - Misrouting

- **Buffered**
  - Store-and-forward
  - Virtual cut-through
  - Wormhole
  - Virtual-channel
Flow Control Protocols

- **Bufferless**
  - Circuit switching
  - Dropping
  - Misrouting

- **Buffered**
  - Store-and-forward
  - Virtual cut-through
  - Wormhole
  - Virtual-channel

Complexity & Efficiency
Circuit Switching

- Form a circuit from source to dest
- Probe to set up path through network
- Reserve all links
- Data sent through links

- Bufferless
Time-space View: Circuit Switching

- Why is this good?
- Why is it not?
Time-space View: Circuit Switching

- Why is this good?
- Why is it not?

April 1, 2015
Time-space View: Circuit Switching

- Why is this good?
- Why is it not?
Time-space View: Circuit Switching

- Why is this good?
- Why is it not?
Time-space View: Circuit Switching

- Why is this good?  
  Simple to implement
- Why is it not?

April 1, 2015
Time-space View: Circuit Switching

• Why is this good? Simple to implement
• Why is it not? Wasteful, increased 3X latency for short packets

April 1, 2015
Sanchez & Emer
Speculative Flow Control: Dropping

- If two things arrive and I don’t have resources, drop one of them
- Flow control protocol on the Internet
Speculative Flow Control: Dropping

- If two things arrive and I don’t have resources, drop one of them
- Flow control protocol on the Internet
Time-space Diagram: Dropping

Disadvantages:
Time-space Diagram: Dropping

Unable to allocate channel 3

Disadvantages:
Time-space Diagram: Dropping

Unable to allocate channel 3

Disadvantages:

Retransmission
Time-space Diagram: Dropping

Unable to allocate channel 3

Disadvantages:
Poor tradeoff of traffic and buffering
Less Simple Flow Control: Misrouting

• If only one message can enter the network at each node, and one message can exit the network at each node, the network can never be congested. Right?
Less Simple Flow Control: Misrouting

- If only one message can enter the network at each node, and one message can exit the network at each node, the network can never be congested. Right?

Wrong! Multiple hops cause congestion
Less Simple Flow Control: Misrouting

- If only one message can enter the network at each node, and one message can exit the network at each node, the network can never be congested. Right?

Wrong! Multiple hops cause congestion

- Philosophy behind misrouting: intentionally route away from congestion
- No need for buffering
Less Simple Flow Control: Misrouting

• If only one message can enter the network at each node, and one message can exit the network at each node, the network can never be congested. Right?

Wrong! Multiple hops cause congestion

• Philosophy behind misrouting: intentionally route away from congestion
• No need for buffering
Less Simple Flow Control: Misrouting

- If only one message can enter the network at each node, and one message can exit the network at each node, the network can never be congested. Right?

Wrong! Multiple hops cause congestion

- Philosophy behind misrouting: intentionally route away from congestion
- No need for buffering
Less Simple Flow Control: Misrouting

• If only one message can enter the network at each node, and one message can exit the network at each node, the network can never be congested. Right?

Wrong! Multiple hops cause congestion

• Philosophy behind misrouting: intentionally route away from congestion

• No need for buffering
Less Simple Flow Control: Misrouting

• If only one message can enter the network at each node, and one message can exit the network at each node, the network can never be congested. Right?

  Wrong! Multiple hops cause congestion

• Philosophy behind misrouting: intentionally route away from congestion

• No need for buffering
Less Simple Flow Control: Misroutting

• If only one message can enter the network at each node, and one message can exit the network at each node, the network can never be congested. Right?

Wrong! Multiple hops cause congestion

• Philosophy behind misrouting: intentionally route away from congestion
• No need for buffering
• Problems?
Less Simple Flow Control: Misrouting

• If only one message can enter the network at each node, and one message can exit the network at each node, the network can never be congested. Right?

Wrong! Multiple hops cause congestion

• Philosophy behind misrouting: intentionally route away from congestion
• No need for buffering
• Problems?

Liveloack: need to guarantee that progress is made
Buffered Routing

• **Link-level flow control:**
  - Given that you can’t drop packets, how to manage the buffers? When can you send stuff forward, when not?

• **Metrics of interest:**
  - Throughput/Latency
  - Buffer utilization (turnaround time)
Techniques for link backpressure
Techniques for link backpressure

• Naïve stall-based (on/off):
  – Can source send or not?
Techniques for link backpressure

• Naïve stall-based (on/off):
  – Can source send or not?

• Sophisticated stall-based (credit-based):
  – How many flits can be sent to the next node?
Techniques for link backpressure

• Naïve stall-based (on/off):
  – Can source send or not?

• Sophisticated stall-based (credit-based):
  – How many flits can be sent to the next node?

• Speculative (ack/nack):
  – Guess can always send, but keep copy
  – Resolve if send was successful (ack/nack)
    • On ack – drop copy
    • On nack - resend
Store-and-Forward (packet-based, no flits)

• **Strategy:**
  – Make intermediate stops and wait until the entire packet has arrived before you move on

• **Advantage:**
  – Other packets can use intermediate links
Time-space View: Store-and-Forward

- Buffering allows packet to wait for channel
- Drawback?
Time-space View: Store-and-Forward

Could be allocated at a much later time without packet dropping

- Buffering allows packet to wait for channel
- Drawback?
Time-space View: Store-and-Forward

- Buffering allows packet to wait for channel
- Drawback? Serialization latency experienced at each hop/channel

Could be allocated at a much later time without packet dropping
Virtual Cut-through (packet-based)

- Why wait till entire message has arrived at each intermediate stop?
- The head flit of the packet can dash off first
- When the head gets blocked, whole packet gets blocked at one intermediate node
- Used in Alpha 21364
Time-space View: Virtual Cut-through

(A) Channel

(B) Channel

• Advantages?

• Disadvantages?
Time-space View: Virtual Cut-through

• Advantages?

• Disadvantages?

(A) No breaks allowed

(B)
Time-space View: Virtual Cut-through

- Advantages?
  - Lower latency

- Disadvantages?
Time-space View: Virtual Cut-through

• Advantages?
  Lower latency

• Disadvantages?
  Buffers allocated in packets → large buffers & low utilization
  Channels allocated in packets → unfairness & low utilization
Flit-Buffer Flow Control: Wormhole

- When a packet blocks, just block wherever the pieces (flits) of the message are at that time.

- Operates like cut-through but with channel and buffers allocated to flits rather than packets
  - Channel state (virtual channel) allocated to packet so body flits can follow head flit
Time-space View: Wormhole

- Advantages?
- Disadvantages?
Time-space View: Wormhole

- Advantages?
- Disadvantages?

Smaller amount of buffer space required
Time-space View: Wormhole

- Advantages?
  - Smaller amount of buffer space required

- Disadvantages?
  - May block a channel mid-packet, another packet cannot use bandwidth
Virtual-Channel (VC) Flow Control

• When a message blocks, instead of holding on to links so others can’t use them, hold on to virtual links

• Multiple queues in buffer storage
  – Lanes on the highway

• Virtual channel can be thought of as channel state and flit buffers
Time-space View: Virtual-Channel

- Advantages?
- Disadvantages?
Time-space View: Virtual-Channel

Advantages?
Significantly reduces blocking in VC buffer

Disadvantages?
Time-space View: Virtual-Channel

- Advantages?
  - Significantly reduces blocking

- Disadvantages?
  - More complex router, fair VC allocation required
Next Time:

Router (Switch) Microarchitecture
Routing Algorithms