EECS 570
Lecture 19
Interconnects: Intro
Winter 2017
Prof. Thomas Wenisch

http://www.eecs.umich.edu/courses/eecs570/

Announcements

• Programming Assignment 2
  ❑ Tonight at 11:59pm

• Final Exam
  ❑ 4/21 4:00-6:00pm

• Final Project
  ❑ Report due & Poster session 4/24 10:30-12:30pm
Readings

For today:
- Jerger & Peh. Interconnection Networks Ch. 3

For Wednesday 3/29:
- Jerger & Peh. Interconnection Networks Ch. 4
Current Hardware Limitations

- **Complexity**
  - Subtle races and numerous transient states in the protocol
    - Hard to extend for optimizations

- **Storage overhead**
  - Directory overhead for sharer lists (makes up for new bits at ~20 cores)

- **Performance and power inefficiencies**
  - Invalidation, ack messages
    - Indirection through directory
  - False sharing (cache-line based coherence)
    - Bandwidth waste (cache-line based communication)
Insights

1. Traditional directory must be updated at every transfer
   DeNovoD can copy valid data around freely

2. Traditional systems send cache line at a time
   DeNovoD uses regions to transfer only relevant data
   Effect of AoS-to-SoA transformation w/o programmer/compiler
Flexible, Direct Communication

L1 of Core 1

... R X0 V Y0 V Z0
... R X1 V Y1 V Z1
... R X2 V Y2 V Z2
... I X3 V Y3 V Z3
... I X4 V Y4 V Z4
... I X5 V Y5 V Z5

Shared L2

LD X3

L1 of Core 2

... I X0 V Y0 V Z0
... I X1 V Y1 V Z1
... I X2 V Y2 V Z2
... R X3 V Y3 V Z3
... R X4 V Y4 V Z4
... R X5 V Y5 V Z5

Registered
Valid
Invalid
Flexible, Direct Communication
Current Hardware Limitations

- **Complexity**
  - Subtle races and numerous transient sates in the protocol
  - Hard to extend for optimizations

- **Storage overhead**
  - Directory overhead for sharer lists (makes up for new bits at ~20 cores)

- **Performance and power inefficiencies**
  - Invalidation, ack messages
  - Indirection through directory
  - False sharing (cache-line based coherence)
  - Bandwidth waste (cache-line based communication)
Evaluation

- Verification: DeNovoD vs. MESI word w/ Murphi model checker [Hipec’15]
  - Correctness
    - Six bugs in MESI protocol: Difficult to find and fix
    - Three bugs in DeNovoD protocol: Simple to fix
  - Complexity
    - 15x fewer reachable states for DeNovoD
    - 20x difference in the runtime
- Performance: Simics + GEMS + Garnet
  - 64 cores, simple in-order core model
- Workloads
  - FFT, LU, Barnes-Hut, and radix from SPLASH-2
  - bodytrack and fluidanimate from PARSEC 2.1
  - kd-Tree (two versions) [HPG 09]
DeNovoD is comparable to or better than MESI
DeNovoD + opts shows 32% lower memory stalls vs. MESI (max 77%)
DeNovoD has 36% less traffic than MESI (max 71%)
Relaxing Software Constraints for DeNovoD

• DeNovoND adds disciplined locks [ASPLOS’13, Top Picks’14]
  – When to self-invalidate: at lock acquire
  – What data to self-invalidate: dynamically collect modified data signatures
  – Special hardware support for locks

• DeNovoSync adds arbitrary synchronization [ASPLOS’15]
  – Software requirement: Data-race-free
    • Distinguish synchronization vs. data accesses to hardware
    • Obeyed by C++, C, Java, ...
    • Optional software information for data consistency performance
  – Semantics: Sequential consistency
  – Hardware: Register all synch reads,
    HW backoff on read-read race w/ contention
DeNovoSync0 Protocol

- **Key:** Synch read should not return stale data

- **When to self-invalidate synch location?**
  - Every synch read?
  - Every synch read to non-registered state

- **DeNovoSync0 registers (serializes) synch reads**
  - Successive reads hit
  - Updates propagate to readers
• Key: Synch read should not return stale data

• When to self-invalidate synch location?
  – Every synch read?
  – Every synch read to non-registered state

• DeNovoSync0 registers (serializes) synch reads
  – Successive reads hit
  – Updates propagate to readers
  – BUT many registration transfers for Read-Read races
DeNovoSync = DeNovoSync0 + Hardware Backoff

- **Hardware backoff** to reduce Read-Read races
  - Remote synch read requests = hint for contention
  - Delay next (local) synch read miss for backoff cycles

- Two-level **adaptive counters** for backoff cycles
  - \( B \) = Per-core backoff counter
    - On remote synch read request, \( B \leftarrow B + I \)
  - \( I \) = Per-core increment counter
    - On \( N \)th remote synch read request, \( I \leftarrow I + D \)
      - \( D \) = Default increment value
    - \( N \) determined by system configuration

Read-Read races \( \Rightarrow \) Contention \( \Rightarrow \) Backoff!

More Read-Read races \( \Rightarrow \) More contention \( \Rightarrow \) Backoff longer!

- \( N \) determined by system configuration
For 44 of 48 cases, 22% lower exec time, 58% lower traffic (not shown)

Remaining 4 cases:

Centralized unbalanced barriers: But tree barriers better for MESI too

Heap with array locks: Need dynamic data signatures for self-invalidation
DeNovo rethinks memory hierarchy for disciplined models

- For deterministic codes
  - **Complexity:** no transients, 20X faster to verify, extensible
  - **Storage overhead:** no directory overhead
  - **Performance, power:** No inv/acks, false sharing, indirection, …
    Up to 77% lower memory stall time, up to 71% lower traffic

- Benefits even for non-determinism and arbitrary synchronization
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 multicomputer 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)

Number of devices interconnected

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
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
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. Request broadcast
3. Send Data

1. Read Cache miss
2. Directory receives request
3. Send Data
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

Status | Addr | Data
--- | --- | ---

MSHRs

Message Format and Send

To network

Dest | RdReq | Addr
--- | --- | ---
Dest | Writeback | Addr | Data
Dest | Reply | Addr | Data

Message Receive

From network

RdReply | Addr | Data
--- | --- | ---
Request | Addr
WriteAck | Addr

Lecture 19 Slide 46
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

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>

To network
Message Format and Send

Memory Controller

Off-chip memory
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 \textit{a priori}
  - \textbf{Automated} design process
  - \textbf{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

Application

Codesign simulation

Input traffic model

Constraint graph
Comm graph

NoC Area models

NoC Power models

User objectives: power, hop delay

Constraints: area, power, hop delay, wire length

NoC Area models

Topology Synthesis
Includes:
Floorplanner NoC Router

SunFloor

System specs:

Platform Generation
(xpipes-Compiler)

NoC Component library

IP Core models

Synthesis

IP Core models

FPGA Emulation

Placement and Routing

To fab

Floorplanning specifications

Area, power characterization

User objectives:
power, hop delay

Constraints:
area, power, hop delay, wire length

NoC Area models

NoC Power models

SystemC code

RTL Architectural Simulation

Synthesis

FPGA Emulation

Placement and Routing

To fab

Floorplanning specifications

Area, power characterization
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