High-Performance On-Chip Interconnects for Emerging SoCs
http://tusharkrishna.ece.gatech.edu/teaching/nocs_acaces17/
ACACES Summer School 2017

Lecture 1: Intro + Topology

Tushar Krishna
Assistant Professor
School of Electrical and Computer Engineering
Georgia Institute of Technology

tushar@ece.gatech.edu

Acknowledgment: Some material adapted from Univ of Toronto ECE 1749 H (N Jerger) and Cornell University ECE 5970 (C Batten)
Introductions!

**TUSHAR KRISHNA**  
Assistant Professor

School of ECE  
Georgia Institute of Technology  
Atlanta, GA 30332, USA

EMAIL: tushar@ece.gatech.edu  
WEB: http://tusharkrishna.ece.gatech.edu

**Background**
- PhD from MIT in EECS (2013)  
- Researcher at Intel (2014-15)  
- Georgia Tech (2015 - present)

**Research Interests**
- Computer Architecture  
- Interconnection Networks  
- Network-on-Chip  
- Deep Learning Accelerators  
- Fog Computing
What is an Interconnection Network?

- Networks **within** a system (collaboration)
- Not the Internet: network **between** systems
What is an Interconnection Network?

Parallel Programming/Software

Massively Parallel Processors

Shared Memory

Memory Consistency

MPI

Infiniband

Myrinet

Datacenters and HPC

Massive Memory

AMBA Bus

Cache Coherence Protocol

System-on-Chip

On-Chip Microarchitecture

Many-core

Mesh

High-Radix

NIC

Optical Waveguides

Equalized Link

CMOS Driver

Repeate Link

Circuits

Memory Consistency

Parallel Programming/Software

Datacenters and HPC
What is an Interconnection Network?

- Interconnection Networks connect processors and memory elements within and across computers.
What is an Interconnection Network?

- **Application:** Ideally wants low-latency, high-bandwidth, dedicated channels between processors and memory

- **Technology:** Dedicated channels too expensive in terms of area and power
What is an Interconnection Network?

- **Interconnection Network**: A programmable system that transports data between terminals over a set of shared physical channels.
Interconnection Networks

Key Design Principles

- Transfer maximum amount of information (high bandwidth) within the least amount of time (low latency) so as to not bottleneck the system.

- Efficiently utilize shared but scarce resources (buffers, links, logic) to reduce area and power.
Interconnection Network Domains

Source: Hennessy and Patterson, 5th Edition, Appendix F
Why do On-Chip Networks matter?

Chinese 260-core processors ShenWei SW26010 enabled supercomputer Sunway TaihuLight be the most productive in the world.

IBM reveals 'brain-like' chip with 4,096 cores

Meet KiloCore, a 1,000-core processor so efficient it could run on a AA battery. This monstrous CPU is 100 times more power-efficient than today's laptops.

IBM pushes silicon photonics with on-chip optics

Big Blue researchers have figured out how to use standard manufacturing processes to make chips with built-in optical links that can transfer 25 gigabits of data per second.
Why study NoCs now?

Data partially collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond
Why study NoCs now?
NoCs are critical for performance

Early designs were buses and point-to-point

DOES NOT SCALE!!!
Agenda

- Course Motivation
- Course Logistics
- Introduction to NoCs
- Topology
Course Logistics

<table>
<thead>
<tr>
<th>Day</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>July 10 (Monday)</td>
<td>Intro + Topology</td>
</tr>
<tr>
<td>July 11 (Tuesday)</td>
<td>Routing</td>
</tr>
<tr>
<td>July 12 (Wednesday)</td>
<td>No Class [poster session]</td>
</tr>
<tr>
<td>July 13 (Thursday)</td>
<td>Flow-Control</td>
</tr>
<tr>
<td>July 14 (Friday)</td>
<td>Microarch + Case Studies</td>
</tr>
</tbody>
</table>

Course Website

http://tusharkrishna.ece.gatech.edu/teaching/nocs_acaces17/

All slides and additional paper readings will be posted here!

Course Textbook (recommended but not necessary)

On-Chip Networks, Second Edition
Natalie Enright Jerger, Tushar Krishna, Li-Shiuan Peh

Agenda

- Course Motivation
- Course Logistics
- Introduction to NoCs
- Topology
Introduction to NoCs

On-Chip Network

- Core + L1$
- Core + L1$
- Core + L1$
- Core + L1$
- Core + L1$
- Core + L1$

L2$
Role of the Network-on-Chip

- “Shared Memory” Systems
  - Transport Cache Lines and Cache Coherence Messages between caches and memory controller(s) in shared memory CMPs

- “Message Passing” Systems
  - Transfer data between IP blocks in MPSoCs (Multi-Processor System on Chip)
Example of Traffic Flows

On-Chip Network

If network delay = 10 cycles, controller delay = 5 cycles, how many cycles before data arrives?
“Tile”

Core will not be shown explicitly in the most of the slides. Only the routers will be.
Network Architecture

- Topology
- Routing
- Flow Control
- Router Microarchitecture
Topology: How to connect the nodes with links

~Road Network
Routing: Which path should a message take

~Series of road segments from source to destination
Flow Control:
When does the message stop/proceed

~Traffic Signals / Stop signs at end of each road segment
Design of traffic intersection (number of lanes, algorithm for turning red/green)
Examples

**Oracle SPARC T5 (2013)**
16 multi-threaded cores and 8 L2 banks connected by a **Crossbar NoC**

**IBM Cell (2005)**
1 general purpose and 8 special purpose engines connected by a **Ring NoC**

**Intel SCC (2009)**
24 tiles with 2 cores each connected by a **Mesh NoC**
What makes network design challenging (and interesting)

- Network resources are distributed with almost no centralized control
- Traffic is (often) unpredictable
  - Why?
    - Mapping of tasks to core
    - Memory layout of data
    - Cache sizes, policies
    - Data sharing
    - ...

- Surprisingly easy for the network to become the bottleneck
Agenda

- Course Motivation
- Course Logistics
- Introduction to NoCs
- Topology
How to connect the nodes with links

~Road Network

[Map of Rome showing nodes and links]
Topology Overview

- Often the first step in network design

- Significant impact on network cost-performance
  - Determines implementation complexity, i.e., **cost**
    - number of routers and links
    - router degree (i.e., ports)
    - ease of layout
  - Determines application **performance**
    - number of hops → latency and energy consumption
    - maximum throughput
How to select a topology?

Application’s Task Communication Graph

Vertices - tasks
Edges - communication

Network Topology Graph

Vertices - cores
Edges - links

Best topology?

Problems?

Cannot change algorithm
Cannot change mapping
Cannot adapt to data-dependent load imbalance in application
Layout/packaging issue with long wires and high-node degree

Topology is fixed at design-time.
Benefits to being regular and flexible
Let us define some design-time metrics

- **Degree** – number of ports at a node
  - Proxy for area/energy cost

- **Bisection Bandwidth** - bandwidth crossing a minimal cut that divides the network in half
  - \((\text{Min # channels crossing two halves}) \times (\text{BW of each channel})\)
  - Proxy for peak bandwidth
    - Can be misleading as it does not account for routing and flow control efficiency
    - At this stage, we assume **ideal routing** (perfect load balancing) and **ideal flow control** (no idle cycles on any channel)

- **Diameter** – maximum routing distance (number of links in shortest route)
  - Proxy for latency
Some run-time metrics

- **Hop count (or routing distance)**
  - Number of hops between a communicating pair
  - Depends on application and mapping
  - *Average hop count* or *Average distance*: average hops across all valid routes

- **Channel load**
  - Number of flows passing through a particular link
  - Depends on application and mapping
  - *Maximum channel load* determines throughput

- **Path diversity**
  - Number of shortest paths between a communicating pair
  - Can be exploited by routing algorithm
  - Provides *fault tolerance*
Can you suggest a regular topology (each router with same degree) with smallest possible diameter?

**Trick question :p**
One node. Degree = 0, Diameter = 0.

### Application’s Task Communication Graph
- Vertices - tasks
- Edges - communication
Can you suggest a regular topology (each router with same degree) with **diameter = 1**?

**Application’s Task Communication Graph**

- **Vertices**: tasks
- **Edges**: communication

**Fully Connected**
- **Degree** = ?
- **Bisection BW** = ?

**Challenge**?
- Not scalable!!
- Cannot layout more than 4-6 cores in this manner for area and power reasons
Bus

- **Pros**
  - Cost-effective for small number of nodes
  - Easy to implement snoopy coherence
  - Most multicores with 4-6 cores use Buses

- **Cons**
  - Bandwidth! → Not scalable

Diameter = ?  1
Degree = ?  1
Bisection BW = ?  1
Popular Bus Protocols

- ARM AMBA Bus
  - AHB
  - AXI
  - ACE
  - CHI
- IBM Core Connect
- ST Microelectronics STBus

How to increase bus bandwidth?
- Hierarchical Buses
- Split-buses
Route Packets, Not Wires!

Dally and Towles, DAC 2001

The birth of “on-chip” switched networks
Topology Classification

- **Direct**
  - Each router (switch) is associated with a terminal node
  - All routers are sources and destinations of traffic
  - Example: Ring, Mesh, Torus
    - Most on-chip networks use direct topologies

- **Indirect**
  - Routers (switches) are distinct from terminal nodes
  - Terminal nodes can source / sink traffic
  - Intermediate nodes switch traffic
  - Examples: Crossbar, Butterfly, Clos, Omega, Benes, ...
    - Used in off-chip HPC networks
Ring and Torus

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

8-ary 1-cube

4-ary 2-cube
Ring

- **Pros**
  - Cheap: $O(N)$ cost
  - Used in most multicores today

- **Cons**
  - High latency
  - Difficult to scale – bisection bandwidth remains constant
  - No path diversity
    - 1 shortest path from A to B

**Diameter?** N/2  
**Avg Distance?** N/4  
**Bisection BW?** 2  
**Degree?** 2
Mesh

- **Pros**
  - $O(N)$ cost
  - Easy to layout on-chip: regular and equal-length links
  - Path diversity
    - 3 shortest paths from A to B

- **Cons**
  - Not symmetric on edges
  - Performance sensitive to placement on edge vs. middle
  - Different degrees for edge vs. middle routers
  - Blocking, i.e., certain paths can block others (unlike crossbar)

Diameter? $2(\sqrt{N}-1)$
Bisection BW? $\sqrt{N}$
Degree? 4
Torus

- **Pros**
  - \( O(N) \) cost
  - Exploit locality for near-neighbor traffic
  - High path diversity
    - 6 shortest paths from A to B
  - Edge symmetric
    - good for load balancing
    - Same router degree

- **Cons**
  - Unequal link lengths
  - Harder to layout

Diameter? \( \sqrt{N} \)
Bisection BW? \( 2\sqrt{N} \)
Degree? \( 4 \)
Folded Torus

Easier to layout

Is there any con compared to the mesh?

*All channels have double the length*
Multi-dimensional topologies

- Used in Supercomputers, Datacenters, and other off-chip System Area Networks

- Example:

2,3,4-ary 3 Mesh
Crossbar

- **Pros**
  - Every node connected to all others (**non-blocking**)
  - Low latency and high bandwidth
  - Used by GPUs

- **Cons**
  - Area and Power goes up quadratically (**O(N^2)** cost)
  - Expensive to layout
  - Difficult to arbitrate

- Diameter = ?
- Degree = ?
- Bisection BW = ?

---

Switch
Run-Time Metrics

- Hop Count
  - Latency
- Maximum Channel Load
  - Throughput

We will consider Uniform Random Traffic
Hop Count

9-ary 1 cube

Max = 4
Avg = 2.22

3-ary 2 cube

Max = 2
Avg = 1.33

3-ary 2 mesh

Max = 4
Avg = 1.77

k-ary n cube

\[ H_{avg} = \begin{cases} \frac{nk}{4} & \text{k even} \\ n\left(\frac{k}{4} - \frac{1}{4k}\right) & \text{k odd} \end{cases} \]

k-ary n mesh

\[ H_{avg} = \begin{cases} \frac{nk}{3} & \text{k even} \\ n\left(\frac{k}{3} - \frac{1}{3k}\right) & \text{k odd} \end{cases} \]
Network Latency

\[ T_N = (t_r + t_l) \times H + T_c + T_s \]

- **Router** \( t_r \)
- **Link** \( t_l \)
- **Hops** \( H \)
- **Contention** \( T_c \)

**Symbols**:
- \( T_N \): Network Latency
- \( t_r \): Router Latency
- \( t_l \): Link Latency
- \( H \): Hops
- \( T_c \): Contention Latency
- \( T_s \): Serialization Latency

**Design Time (traffic)**
- **Run Time**
How to reduce hop count?

- Low-diameter topology
  - Challenge?
    - high-radix of each switch

- Some dedicated long-range links
  - High-radix for few switches
    - Popular topologies: Flattened Butterfly, ST Spidergon
  - How to decide where to add long links?
Run-Time Metrics

- Hop Count
  - Latency
- Maximum Channel Load
  - Throughput
Maximum Channel Load

- Identify channel with maximum traffic
- Count total flows through it

Maximum Throughput = 1 / (max channel load)
Maximum Channel Load

- Identify bottleneck channel
  - For uniform random traffic, is the bisection channel

- Suppose each node generates $p$ messages per cycle
  - $4p$ messages per cycle in left ring
  - $2p$ message per cycle will cross to other ring
  - Link can handle one message per cycle
  - So maximum injection rate of $p = \frac{1}{2}$
Maximum Channel Load

- What if Hot Spot Traffic?
  - Suppose every node sends to node G

- Which is the bottleneck channel?
  - Used by A, B, C, D, E, and F to send to G
  - Max Throughput = $1 / 6$
Traffic Patterns

- Historically derived from particular applications of interest
- Important to stress test the network with different patterns
  - Uniform random can make bad topologies look good
- For a particular topology and traffic pattern, one can derive
  - Avg Hop Count (→ Low-Load Latency)
  - Max Channel Load (→ Peak Throughput)

<table>
<thead>
<tr>
<th>Traffic Pattern</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit-Complement</td>
<td>((\bar{y}<em>{k-1}, \bar{y}</em>{k-2}, \ldots, \bar{y}<em>1, \bar{y}<em>0, \bar{x}</em>{k-1}, \bar{x}</em>{k-2}, \ldots, \bar{x}_1, \bar{x}_0))</td>
</tr>
<tr>
<td>Bit-Reverse</td>
<td>((x_0, x_1, \ldots, x_{k-2}, x_{k-1}, y_0, y_1, \ldots, y_{k-2}, y_{k-1}))</td>
</tr>
<tr>
<td>Shuffle</td>
<td>((y_{k-2}, y_{k-3}, \ldots, y_0, x_{k-1}, x_{k-2}, x_{k-3}, \ldots, x_0, y_{k-1}))</td>
</tr>
<tr>
<td>Tornado</td>
<td>((y_{k-1}, y_{k-2}, \ldots, y_1, y_0, x_{k-1+\lceil \frac{k}{2} \rceil-1}, \ldots, x_{\lceil \frac{k}{2} \rceil-1}))</td>
</tr>
<tr>
<td>Transpose</td>
<td>((x_{k-1}, x_{k-2}, \ldots, x_1, x_0, y_{k-1}, y_{k-2}, \ldots, y_1, y_0))</td>
</tr>
<tr>
<td>Uniform Random</td>
<td>(\text{random()})</td>
</tr>
</tbody>
</table>
Saturation Throughput: the injection rate at which latency \( \sim 3x(\text{zero-load latency}) \)

Gaps exist due to inefficiencies in routing and flow-control
Another representation: Injection Rate as a % of “capacity”

For 4x4 Mesh, 100 => 1 flit/node/cycle
For 8x8 Mesh, 100% => 0.5 flits/node/cycle

This representation is better to understand if we are able to achieve the throughput the network was actually designed for.
Which topology should you choose?

- Hard to optimize for everything
  - Desired bandwidth
  - Desired latency

- Physical Constraints
  - Wire budget
    - Indirect topologies popular off-chip
    - On-chip networks often use direct topologies due to wiring constraints
  - Wire layout
    - Topologies should be easy to layout on a planar 2D substrate
  - Router complexity
    - Number of ports
Research in Topology Design

Which topologies are most popular?
- 2D Meshes most popular for CMPs
- Crossbars most popular for GPUs
- 3D Meshes and Torii popular in HPC systems
- Fat Trees popular in Datacenters

Open research questions
- Topologies for many-core systems
- Topologies for heterogeneous SoCs with real-time latency and high-bandwidth requirements
Uniform Random Traffic on a \( k \times k \) Mesh

Zero-load latency? ("Ideal Latency")

\[
T = (H+1) \cdot (t_{\text{router}} + t_{\text{stall\_avg}}) + (H+2) \cdot (t_{\text{wire}}) + T_{\text{ser}}
\]

- \( H \) = number of hops inside network
- \( t_{\text{router}} \) = per-hop router pipeline delay
- \( t_{\text{wire}} \) = per-hop link delay
- \( t_{\text{stall}} \) = per-hop stall delay (due to contention)
- \( T_{\text{ser}} \) = serialization delay

\[
H_{\text{avg}} = \begin{cases} 
\frac{nk}{3} & \text{k even} \\
\frac{n(k^3 - 1)}{3k} & \text{k odd}
\end{cases}
\]

Ideal case: \( t_{\text{router}} = 1, \ t_{\text{wire}} = 1 \)

Let's assume 1-flit packets (\( T_{\text{ser}} = 0 \))

Zero-load \( \Rightarrow \) \( t_{\text{stall\_avg}} \sim 0 \)

Suppose \( k = 8, H_{\text{avg}} = 5.333 \) \( \Rightarrow \) \( T_{\text{zero-load}} = 13.666 \)
Saturation Throughput? ("Ideal Throughput" or Peak Injection Rate)

1 / max channel load

Let's calculate load on one of the bisection links

- \( \frac{k^2}{2} \) nodes on the left.
- Half their messages (\( \frac{k^2}{4} \)) cross the bisection links
- Total \( k \) bisection links from left to right.
- Load on each bisection link = \( \frac{k^2}{4k} = \frac{k}{4} \)
- **Peak Throughput = \( \frac{4}{k} \)**

For \( k = 4 \), peak throughput = 1 flit/node/cycle
For \( k = 8 \) (64-core mesh), peak throughput = \( \frac{1}{2} \) flits / node / cycle