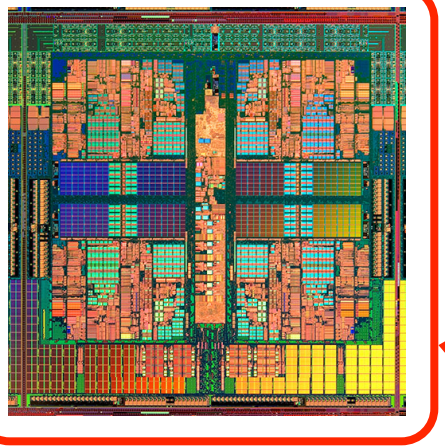
Lecture 26
Interconnection Networks

Suggested reading:

(HP Chapter 7.8)

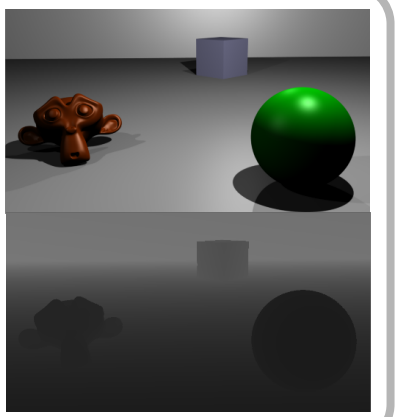
Multicore processors and programming



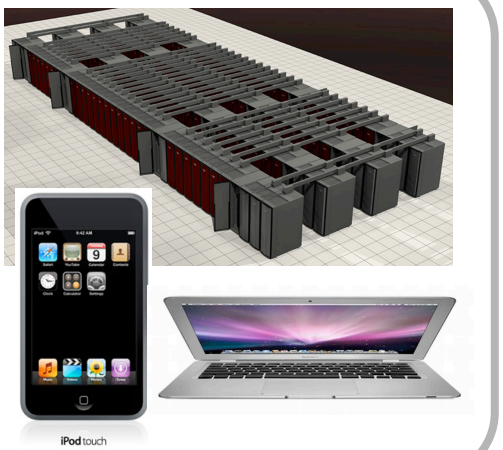
Processor components

Processor comparison

- Explain & articulate why modern microprocessors now have more than one core and how SW must adapt.



Writing more efficient code



The right HW for the right application

```
for i=0; i<5; i++ {
        a = (a*b) + c;
}
```

```
MULT r1,r2,r3    # r1 ← r2*r3
ADD r2,r1,r4     # r2 ← r1+r4
```

110011	000001	000010	000011
001110	000010	000001	000100

HLL code translation

Fundamental lesson(s)

- Additional hardware support is required for parts of a parallel system to communicate with one another
  - (i.e. when one node needs data another has worked on)
  - The overhead associated with communication can actually make part of a program take longer than if the same part were executed serially

Why it's important…

- Communication overhead can/will degrade program performance
  - (Thus, performance improvements you think you'll get by parallelizing your code are not what you actually get)
    - Today we'll talk about another reasons why…
  - Put another way…
    - Assume 1 iteration of a task takes N CCs
    - Parallelizing the task's execution should speed up the total task, but now each iteration may take N+M CCs
      - The M CC overhead can (i) reduce performance gains one might expect and (ii) impact the degree of parallelization that should be employed

Impediments to Parallel Performance

- **Reliability:**
  - Want to achieve high "up time" – especially in non-CMPs
- **Contention for access to shared resources**
  - i.e. multiple accesses to limited # of memory banks may dominate system scalability
- **Programming languages, environments, & methods:**
  - Need simple semantics that can expose computational properties to be exploited by large-scale architectures
- **Algorithms**

Not all problems are parallelizable

$$\text{Speedup} = \frac{1}{\left[1 - \text{Fraction}_{\text{parallelizable}}\right] + \frac{\text{Fraction}_{\text{parallelizable}}}{N}}$$

What if you write good code for 4-core chip and then get an 8-core chip?

- **Cache coherency**
  - P1 writes, P2 can read
    - Protocols can enable $ coherency but add overhead
- **Overhead where no actual processing is done.**

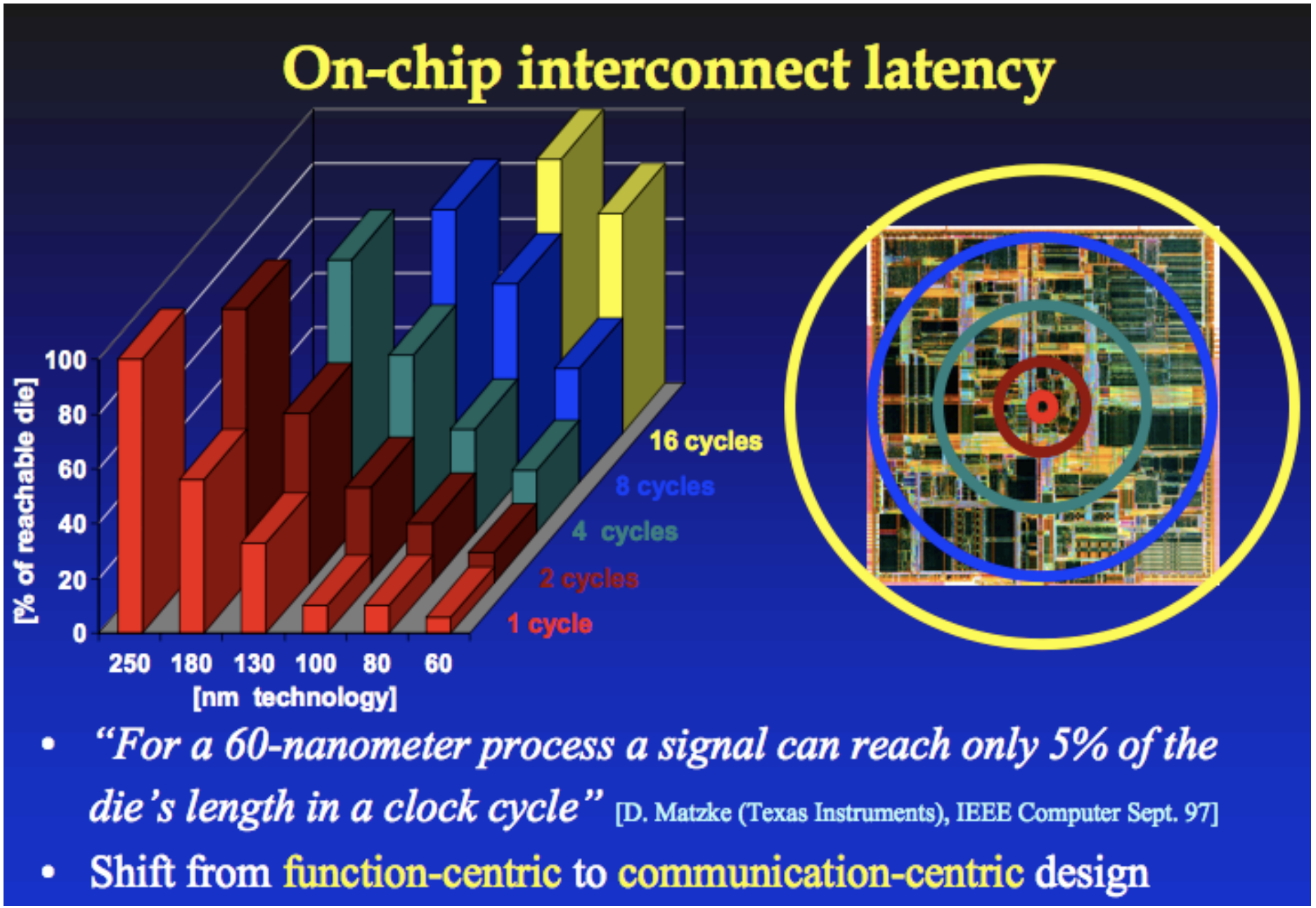
Challenges: Latency★

- …is already a major source of performance degradation
  - Architecture charged with hiding local latency
    - (that's why we talked about registers & caches)
  - Hiding global latency is also task of programmer
    - (I.e. manual resource allocation)
- Today:
  - access to DRAM in 100s of CCs
  - round trip remote access in 1000s of CCs
  - multiple clock cycles to cross chip or to communicate from core-to-core
    - Not "free" as we assumed in send-receive example from L27

We'll talk more quantitatively about this today.

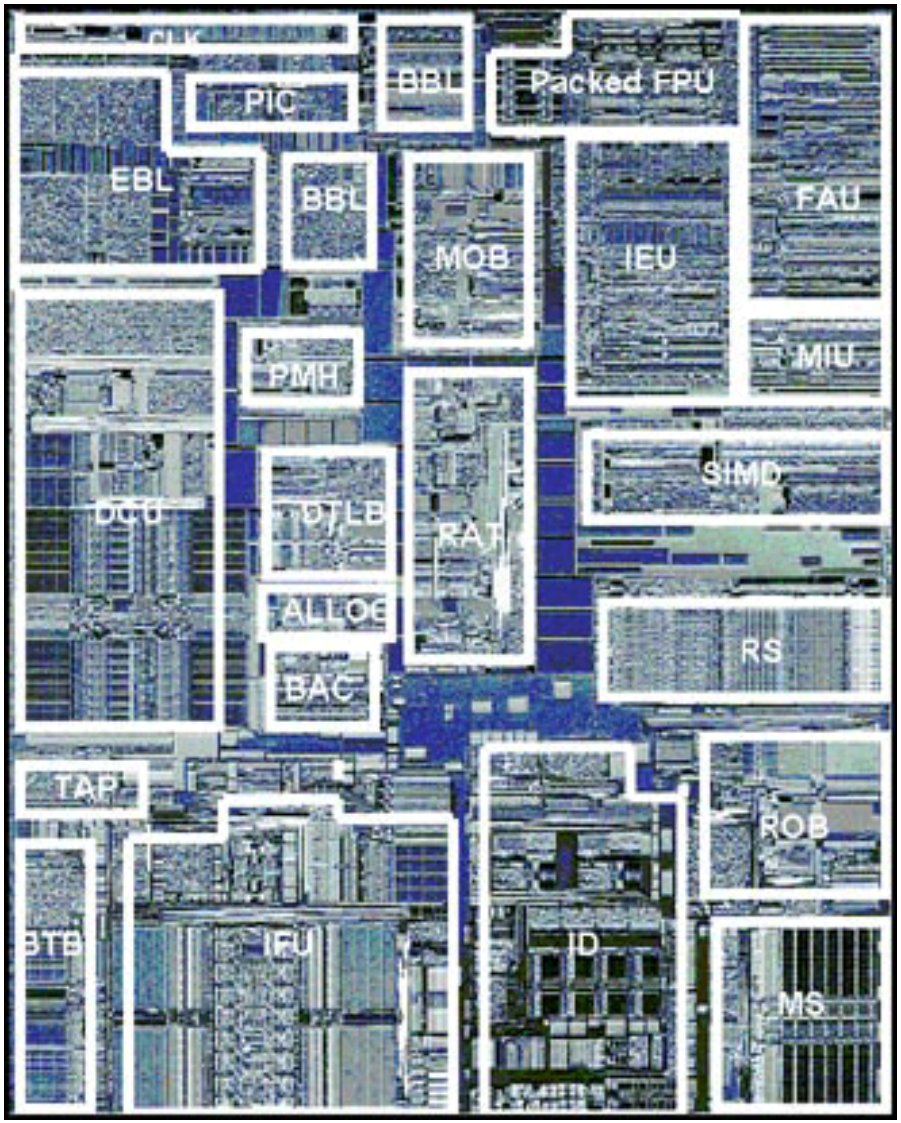
★ Overhead where no actual processing is done.

Some Perspective…



Pentium III Die Photo

Deterministic connections as needed.

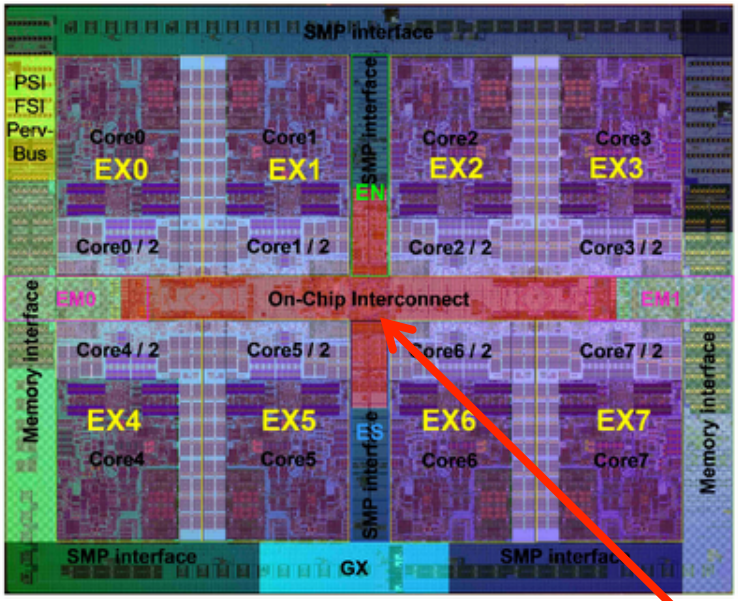


1st Pentium III, Katmai: 9.5 M transistors, 12.3 * 10.4 mm in 0.25-mi. **with 5 layers of aluminum**

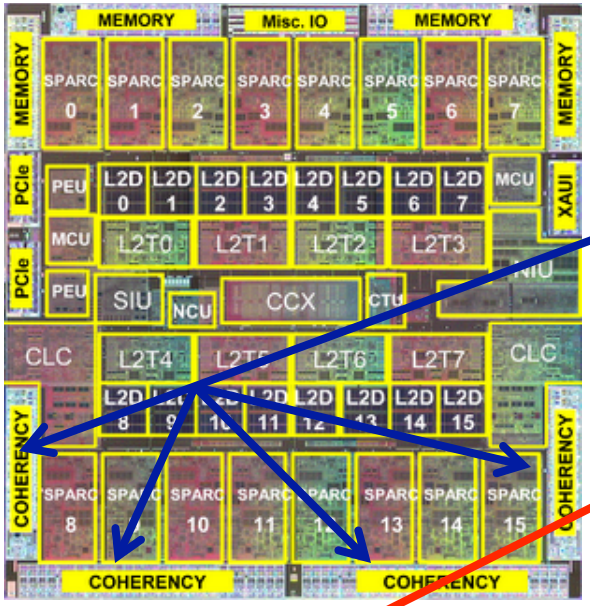
- **EBL/BBL - Bus logic, Front, Back**
- **MOB - Memory Order Buffer**
- **Packed FPU - MMX Fl. Pt. (SSE)**
- **IEU - Integer Execution Unit**
- **FAU - Fl. Pt. Arithmetic Unit**
- **MIU - Memory Interface Unit**
- **DCU - Data Cache Unit**
- **PMH - Page Miss Handler**
- **DTLB - Data TLB**
- **BAC - Branch Address Calculator**
- **RAT – Register Alias Table**
- **SIMD - Packed Fl. Pt.**
- **RS - Reservation Station**
- **BTB - Branch Target Buffer**
- **IFU - Instruction Fetch Unit (+I$)**
- **ID - Instruction Decode**
- **ROB - Reorder Buffer**
- **MS - Micro-instruction Sequencer**

Recent multi-core die photos

(Route packets, not wires?)



http://dx.doi.org/10.1109/ISSCC.2010.5434074



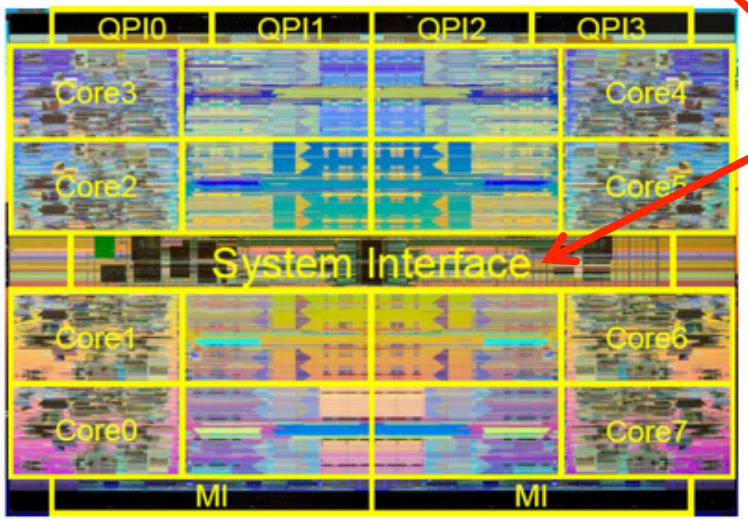
http://dx.doi.org/10.1109/ISSCC.2010.5434030

Likely to see HW support for parallel processor configurations:

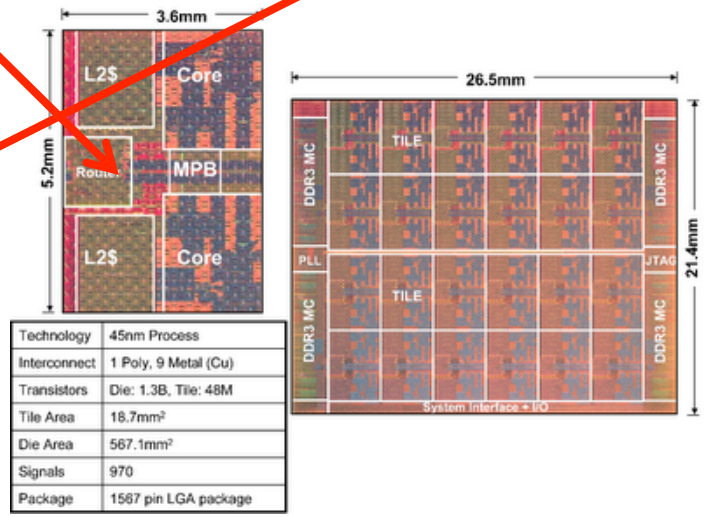
- **Coherency**

+

- **On-chip IC NWs**



http://dx.doi.org/10.1109/ASSCC.2009.5357230



http://dx.doi.org/10.1109/ISSCC.2010.5434077

…takes advantage of 8 voltage and 28 frequency islands to allow independent **DVFS** of cores and mesh. As performance scales, the processor dissipates between 25 W and 125 W. … 567 mm^2 processor on **45 nm CMOS** integrates **48** IA-32 **cores** and 4 DDR3 channels in a **2D-mesh network**. **Cores communicate through message passing** using 384 KB of on-die shared memory. Fine-grain power management

Takeaways from last 2 slides

- Cores communicate with each other
  - (and each others memory)

- Can no longer just realize direct, deterministic connection between processor's functional units

- Fortunately, wide body of work to start with to enable more efficient/reasonable inter-core communication

IMPLEMENTING ON-CHIP INTERCONNECTION NETWORKS

Lot's of history to leverage…

- Lot's of XAN's
  - SAN – system area network
    - Usually connects homogeneous nodes
    - Physical extent small – less than 25 meters – often less
    - Connectivity ranges from 100s to 1000s of nodes
    - Main focus is high bandwidth and low latency
  - LAN – local area network
    - Heterogeneous hosts assumed – designed for generality
    - Physical extent usually within a few hundred kms
    - Connectivity usually in the hundreds of nodes
    - Supported by workstation industry

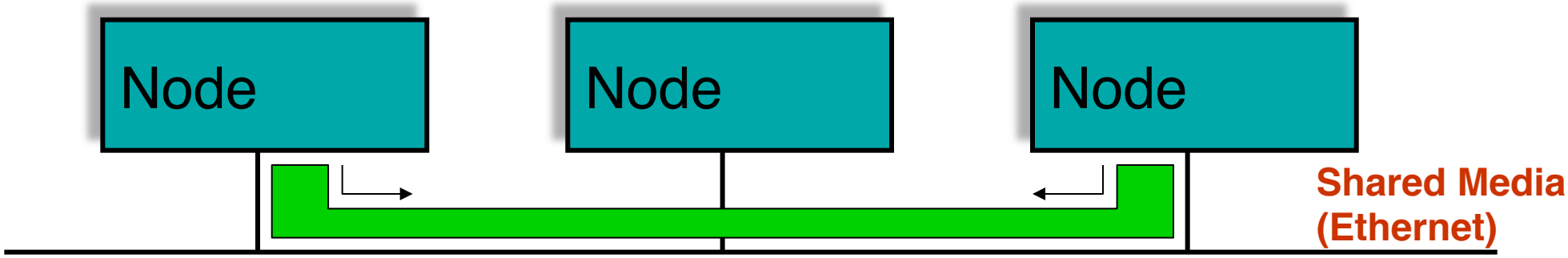
Lot's of history to leverage…

- WAN – wide area network
  - General connectivity for 1000s of heterogeneous nodes
  - High bandwidth (good), high latency (not so good)
  - Physical extent = thousands of kilometers
  - Developed by the telecommunications industry

- **Idea:**
  - Borrow knowledge, lessons learned from these application spaces in design/creation of on-chip networks

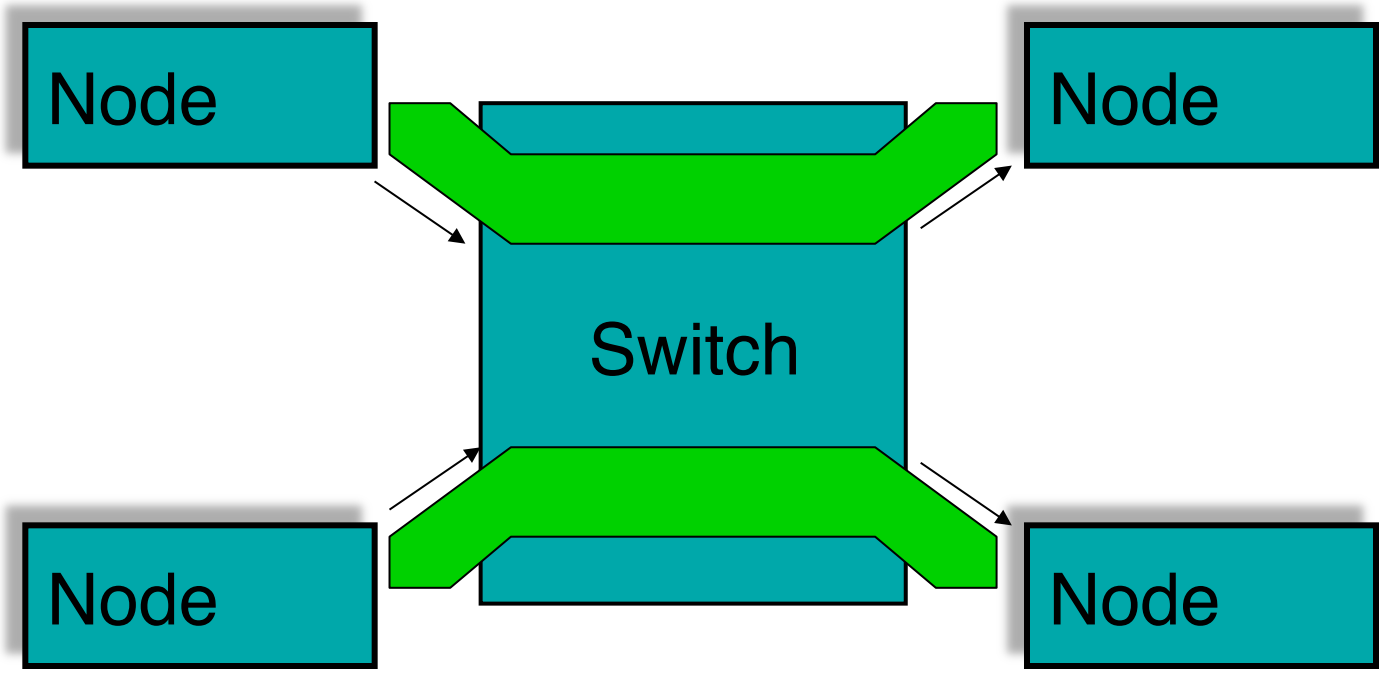
Shared media networks

- **Messages are broadcast everywhere**
  - **Useful for cache coherency**
  - **Not unlike ethernet**



Example… 2 nodes attempt to write to same shared location…

Switched media networks



- Switches introduce overheads
  - But, no time wasted on arbitration and collisions
- Multiple transfers can be in progress if different links used
- Circuit or Packet Switching
  - Circuit switching: end-to-end connections
    - Reserves links for a connection (e.g. phone network)
  - Packet switching: each packet routed separately
    - Links used only when data transferred (e.g. Internet Protocol)

Shared vs. switched media

- **Shared Media** *1*
  - **Broadcast to everyone!**

versus

- **Switched Media (needs real routing)** *2*
  - **Source-based routing: *message* specifies path to the destination**
  - **Virtual Circuit: circuit established from source to destination, message picks circuit to follow**
  - **Destination-based routing: message specifies destination, switch must pick the path**
    - **deterministic: always follow same path**
    - **adaptive: pick different paths to avoid congestion, failures**
    - **randomized routing: pick between several good paths to balance network load**

Switched media: message transmission

- **Store-and-Forward** *1*
  - Switch receives entire packet, then forwards it

versus

- **Wormhole routing** *2*
  - Packet consists of flits (N bytes each)
  - First flit contains header with destination address
  - Switch gets header, decides where to forward
  - Other flits forwarded as they arrive
  - If traffic?
    - Stop the tail when head stops
    - Each flit along the way blocks the a link
    - One busy link creates other busy links (and a traffic jam!)

Switched media: message transmission

- **Cut-Through Routing**
  - In absence of traffic, similar to wormhole…
  - If outgoing link busy…
    - Receive and buffer incoming flits
    - Buffered flits remain until link is free
    - When link free, flits start worming out of the switch
    - Need packet-sized buffer space in each switch
      - (Wormhole routing switch needs to buffer only one flit)

Summary: wormhole vs. cut through

- **Wormhole routing:**
  - **When head of message is blocked, message stays strung out over the network**
    - **Potentially blocking other messages…**
    - **…but needs only buffer the piece of the packet that is sent between switches**

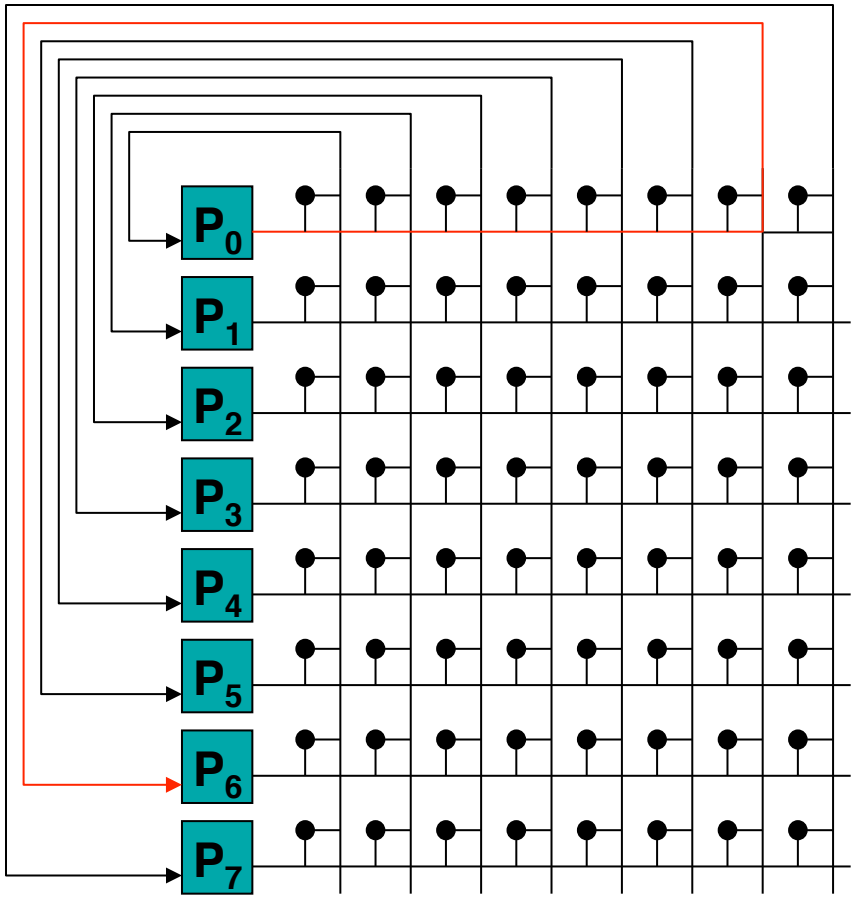
- **Cut through routing**
  - **Lets tail continue when head is blocked**
    - **Whole message is accordian'ed into a single switch**
    - **Requires a buffer large enough to hold the largest packet**

(More connectivity = more hardware – that's harder to implement)

HOW ARE NETWORKS ORGANIZED?

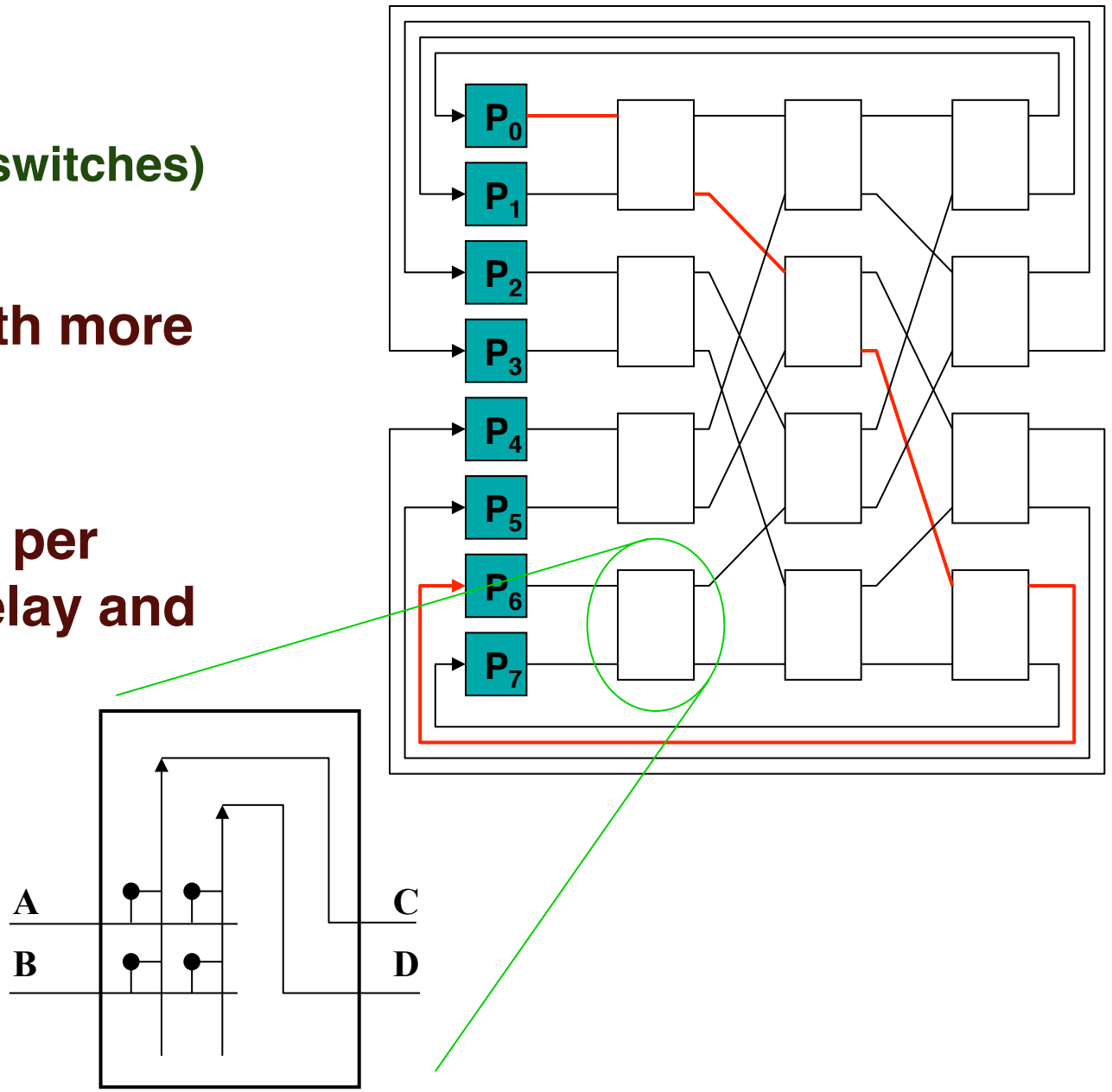
Crossbars

- Crossbars
  - Any node can communicate with another with 1 pass through IC
  - Very low switching delay, no internal contention
  - Complexity grows as square of number of links
    - Cannot have too many links (i.e. 64 in, 64 out)

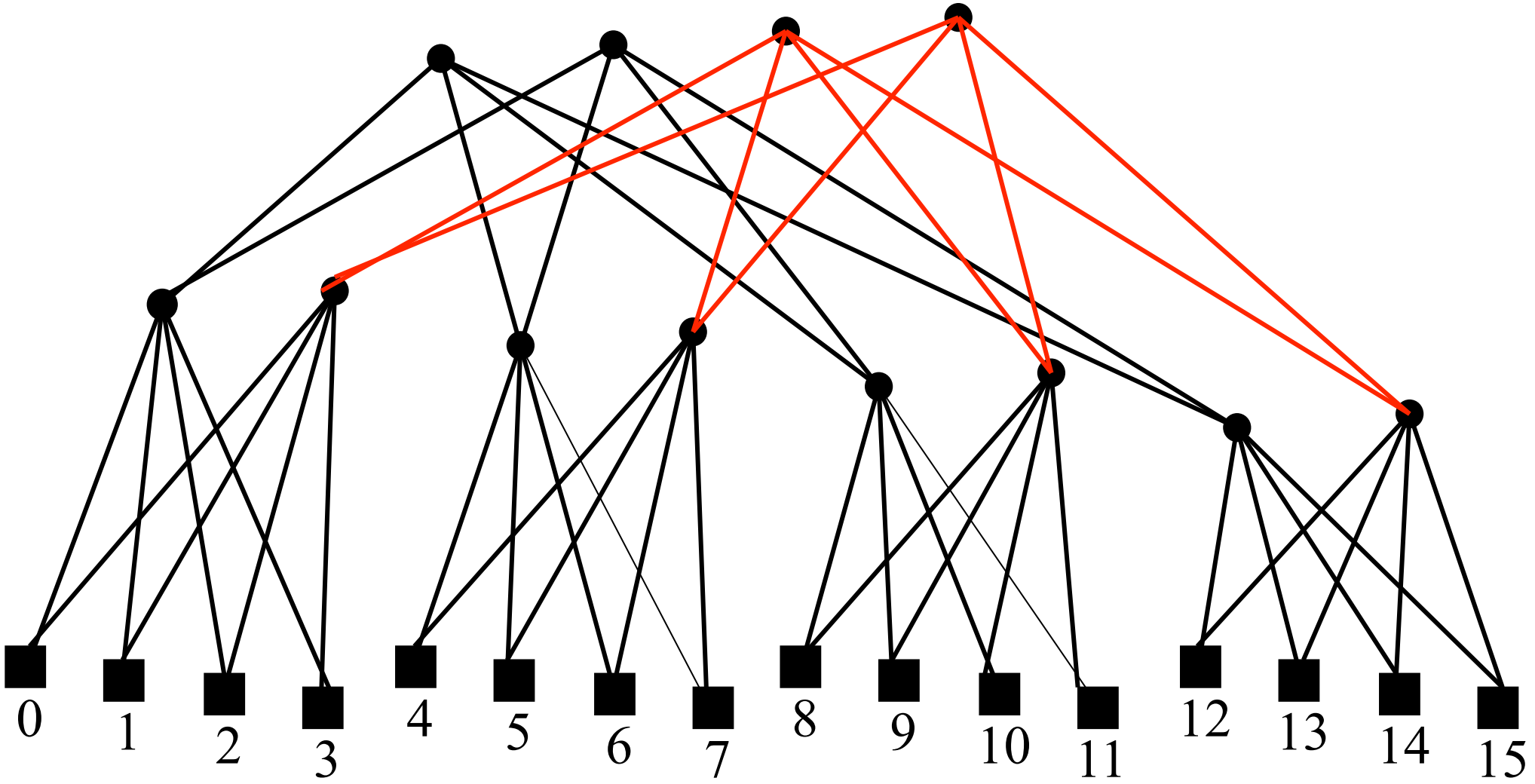


Omega networks

- Omega
  - Uses less HW
    - (n/2 $\log_2 n$ vs. n^2 switches)
  - More contention
  - Build switches with more ports using small crossbars
  - Lower complexity per link, but longer delay and more contention



Tree topologies



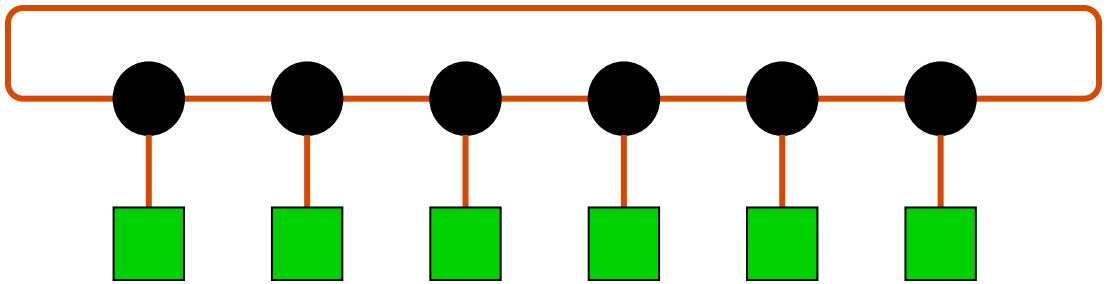
Circles = switches, squares = processor-memory nodes

Higher bandwidth, higher in the tree – match common communication patterns

Ring topologies

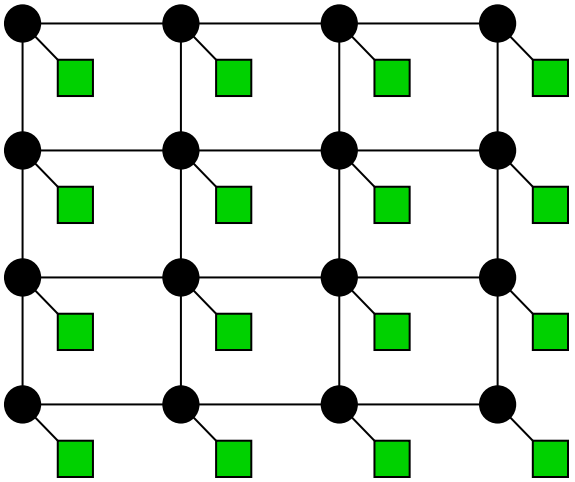
- Small switches are placed at each computer
  - Avoids a full interconnection network
- Disadvantages
  - Some nodes are not directly connected
    - Results in multiple "stops", more overhead
  - Average message must travel through *n*/2 switches
    - (n = # nodes)
- Advantages
  - Rings can have several transfers going at once

Example of a ring topology

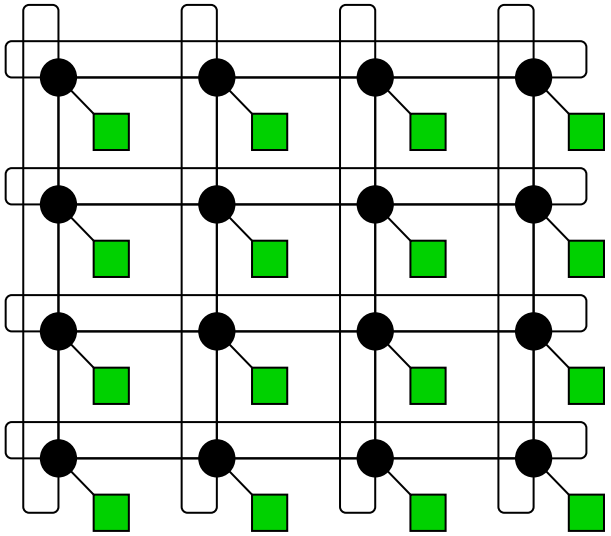


Meshes, tori, hypercubes…

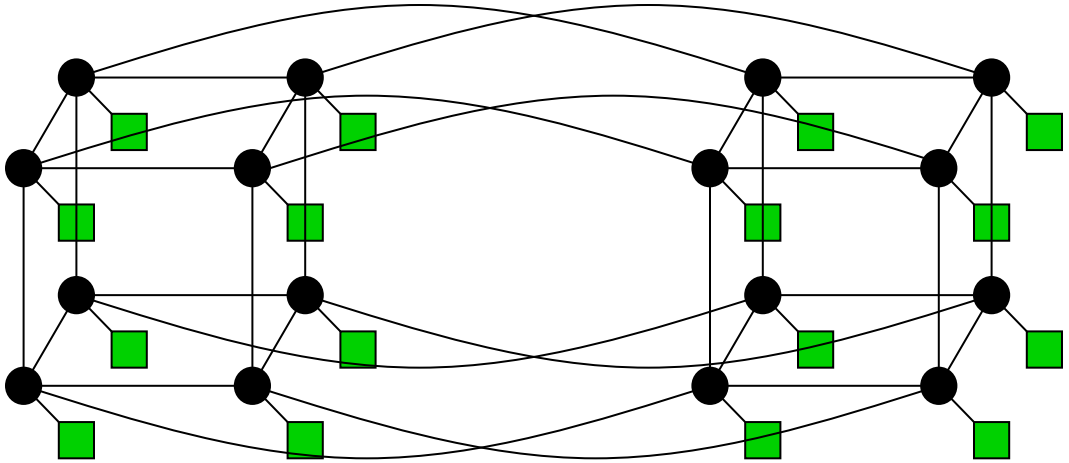
2D grid or mesh of 16 nodes



2D tour of 16 nodes



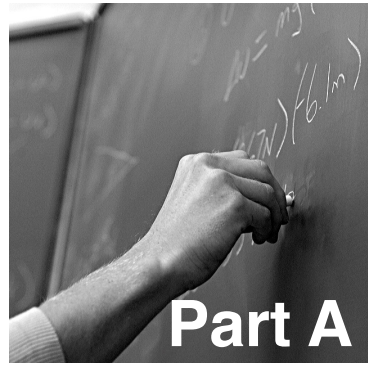
Hypercube tree of 16 nodes (16 = 2^4, so n = 4)



Summarizing thoughts…

- Let's consider crossbar again:
  - A communication link between every switch…
  - An expensive alternative to a ring…
  - Get big performance gains, but big costs as well
    - Usually cost scales by the square of the number of nodes
- High costs led designers to invent "things in between"
  - In other words, topologies between the cost of rings and the performance of fully connected networks
- Whether or not a topology is "good" typically depends on the situation
- For on-chip MPPs, grids, tori, etc. are popular

DISCUSSION: PARTS AND PERFORMANCE OF AN ON-CHIP NETWORK



A 64-CORE CASE STUDY

NW topologies

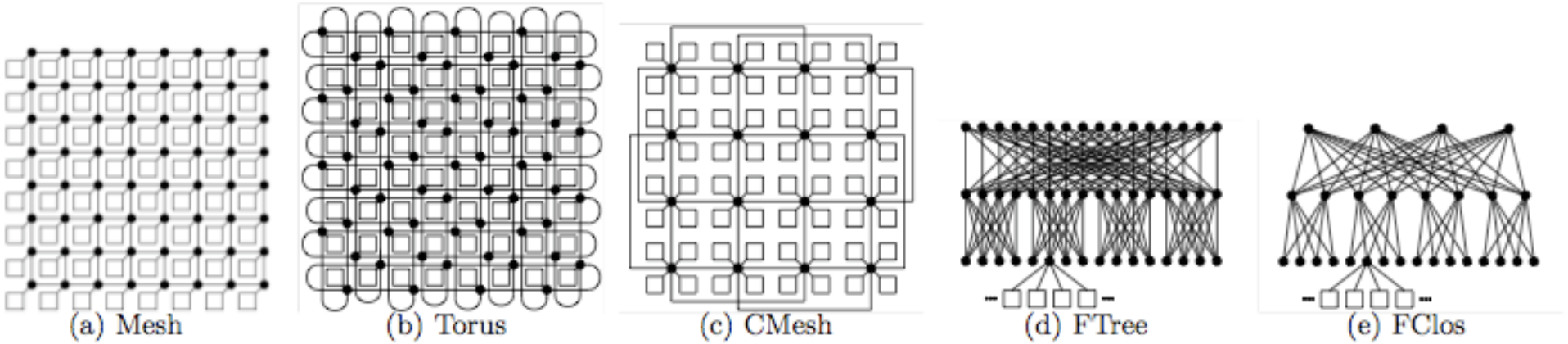


Figure 8: Network Topologies

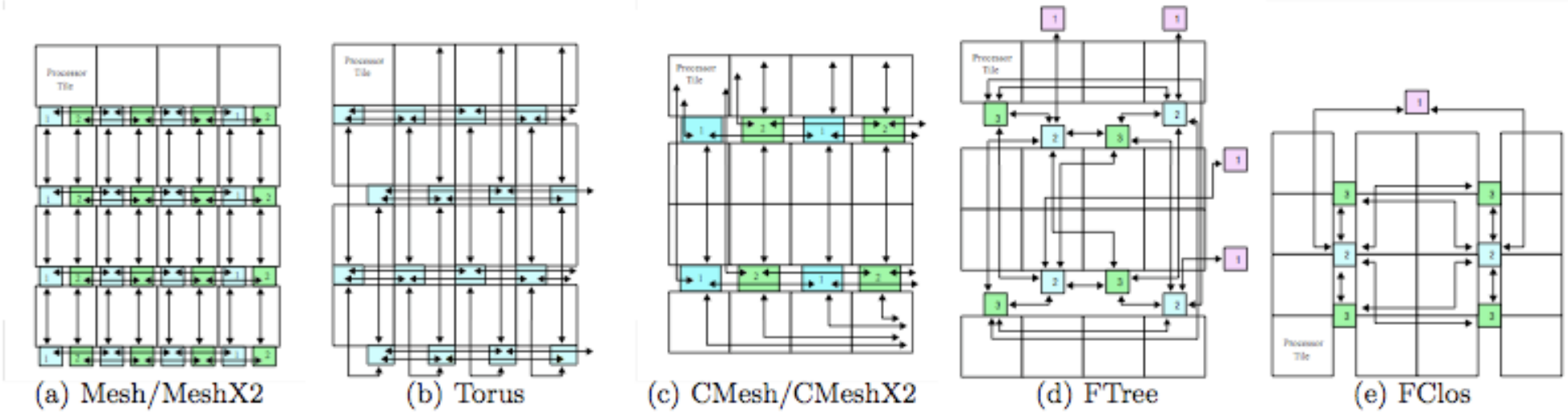
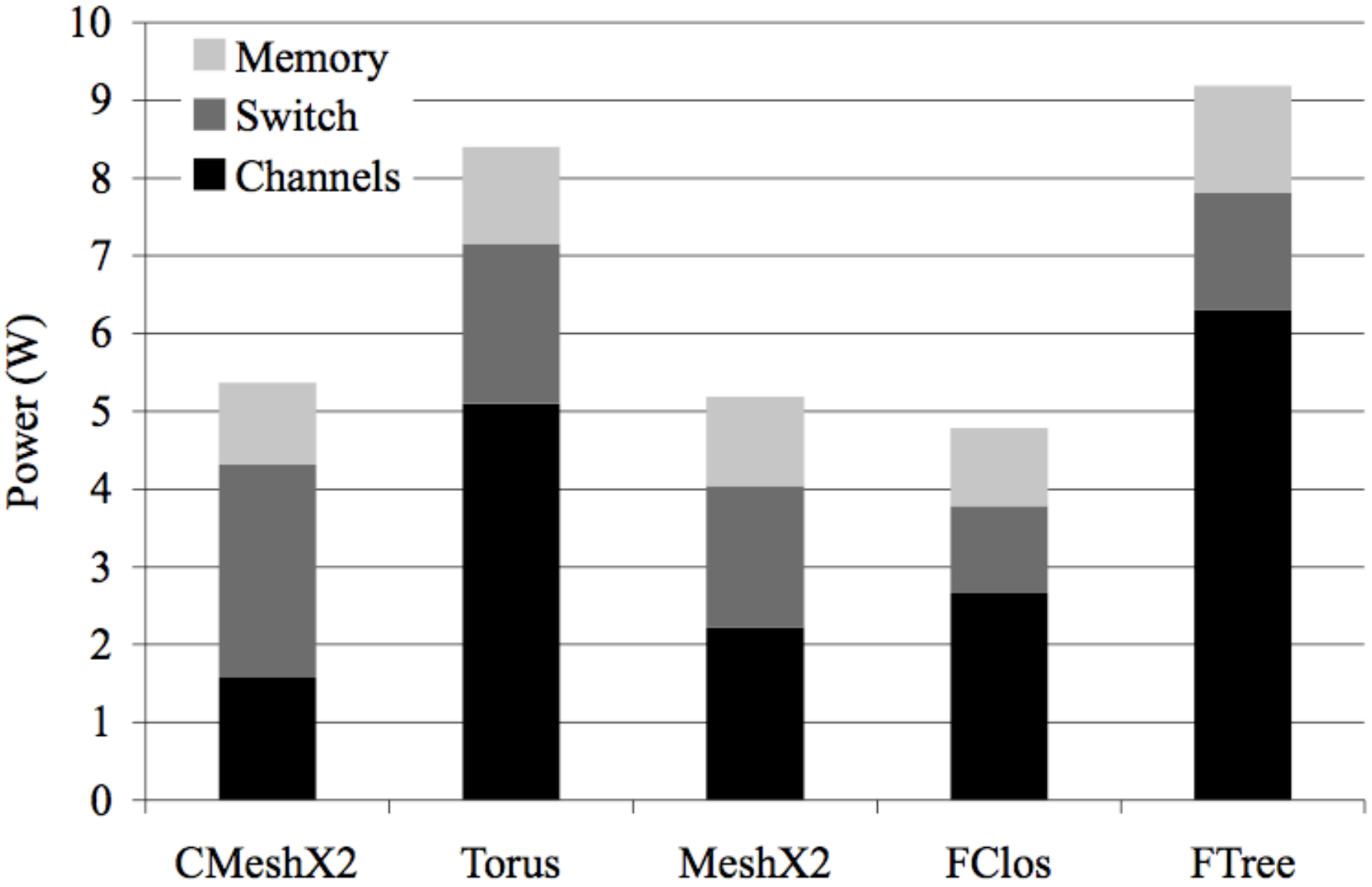
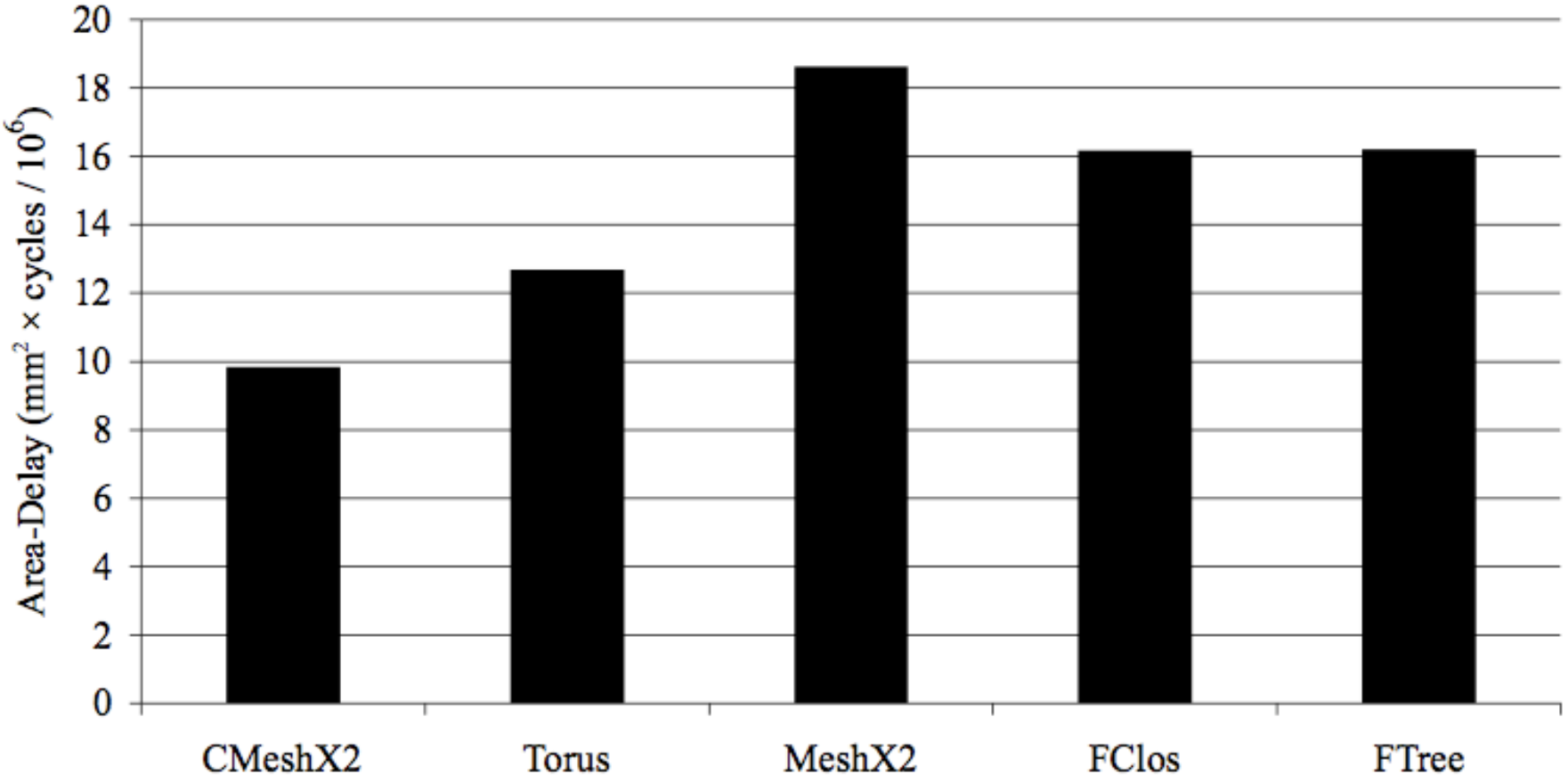


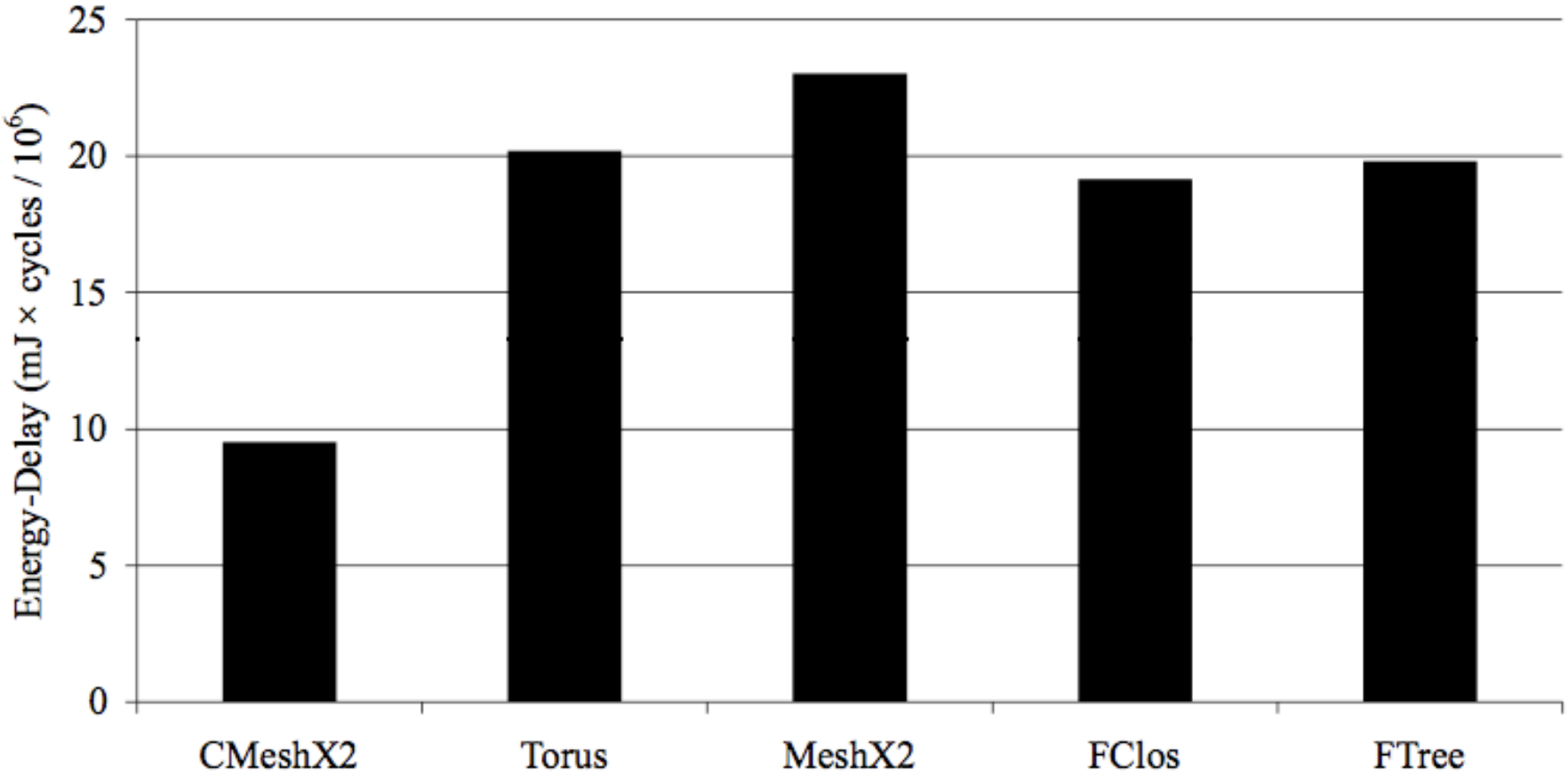
Figure 9: Placement of Routers used to Estimate Area (Lower Left Quadrant)



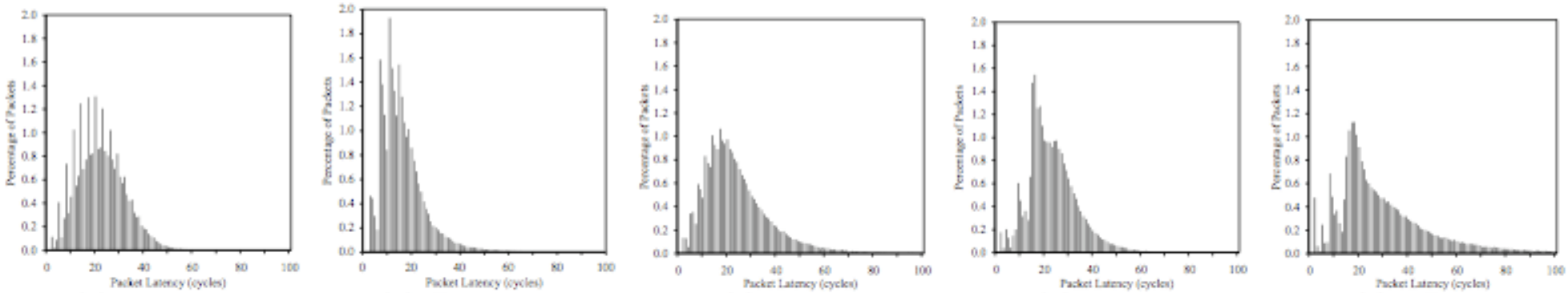
(c) Network Power Dissipation



(d) Area Delay Metric



(e) Energy Delay Metric



(a) MeshX2 (b) CMeshX2 (c) Torus (d) FTree (e) FClos

Figure 11: Workload Packet Latency Distribution for Uniform Random Traffic Pattern

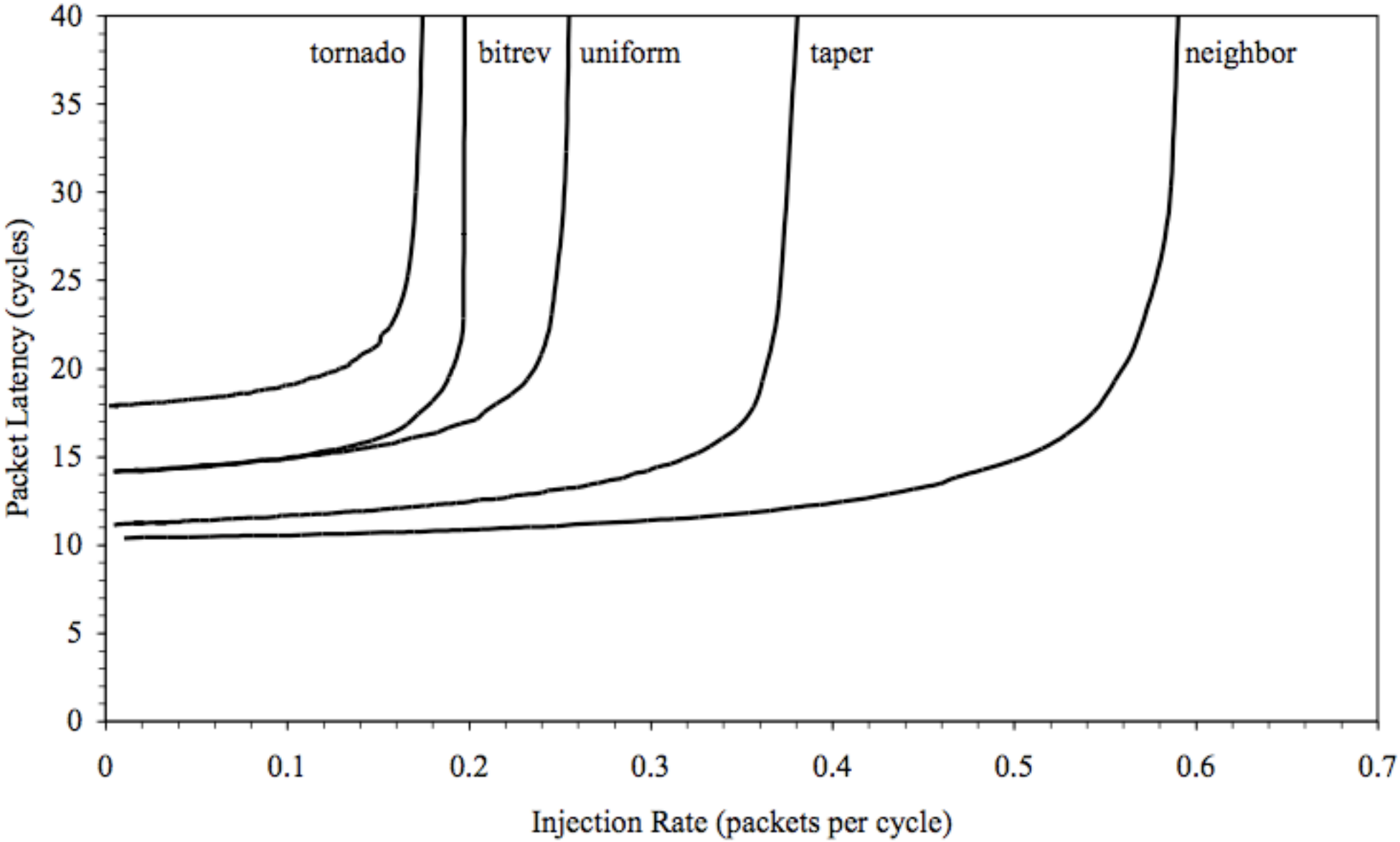


Figure 12: Offered Latency for CMeshX2 Network

A FINAL EXAMPLE