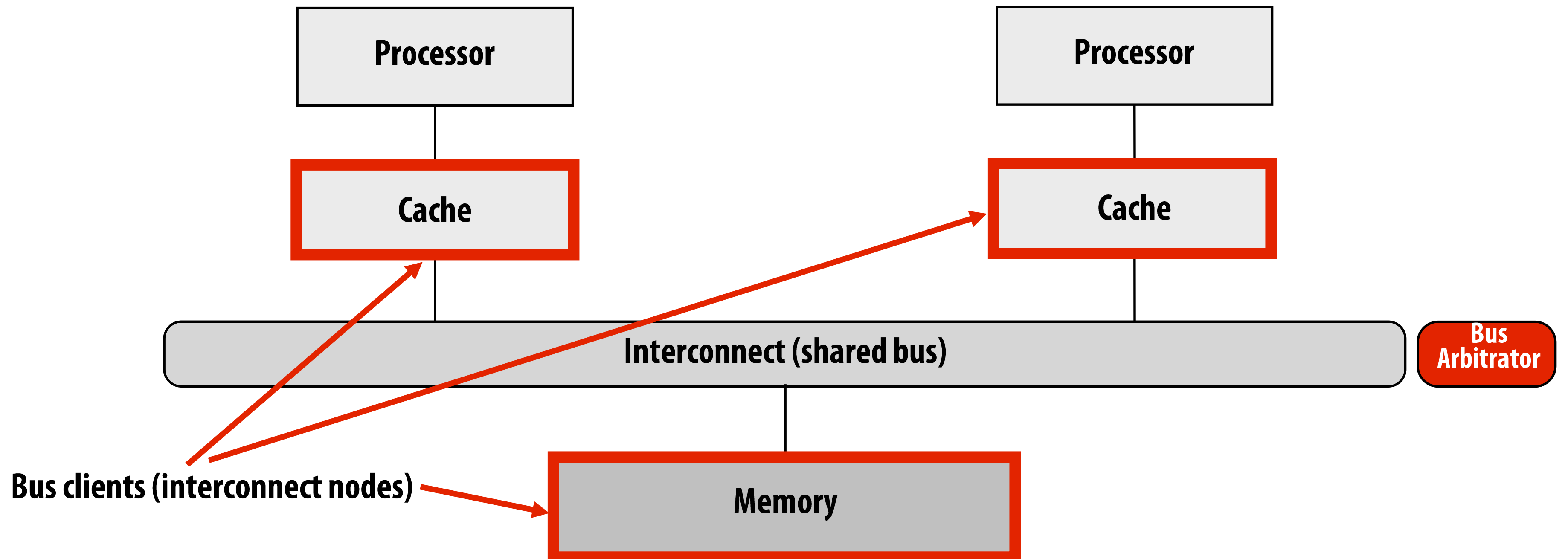
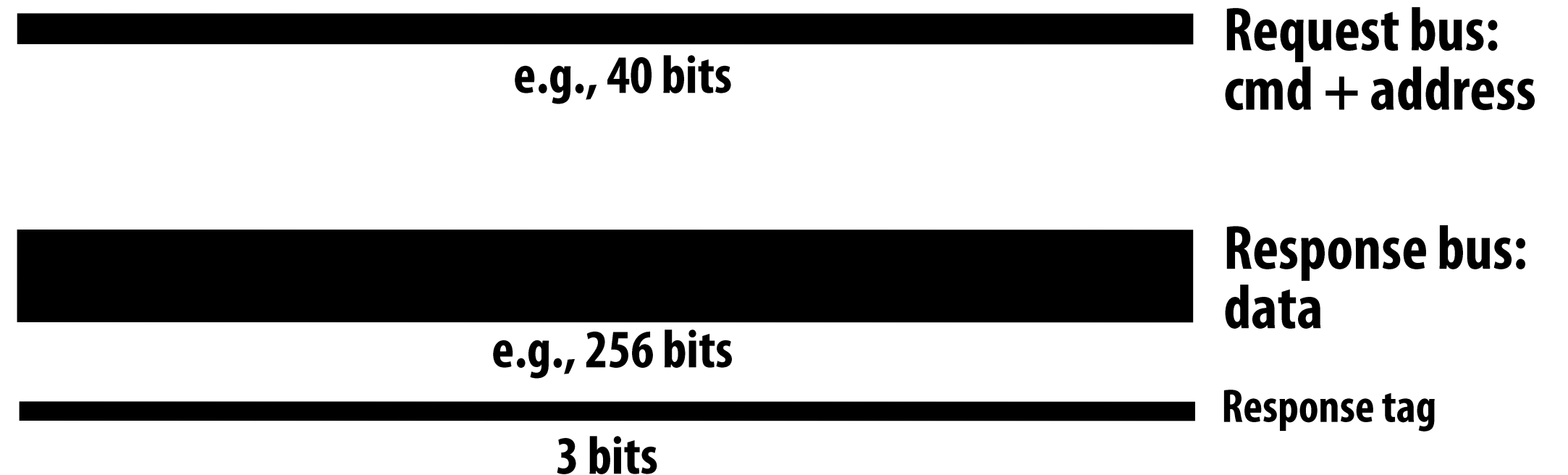


Basic system design from previous lectures

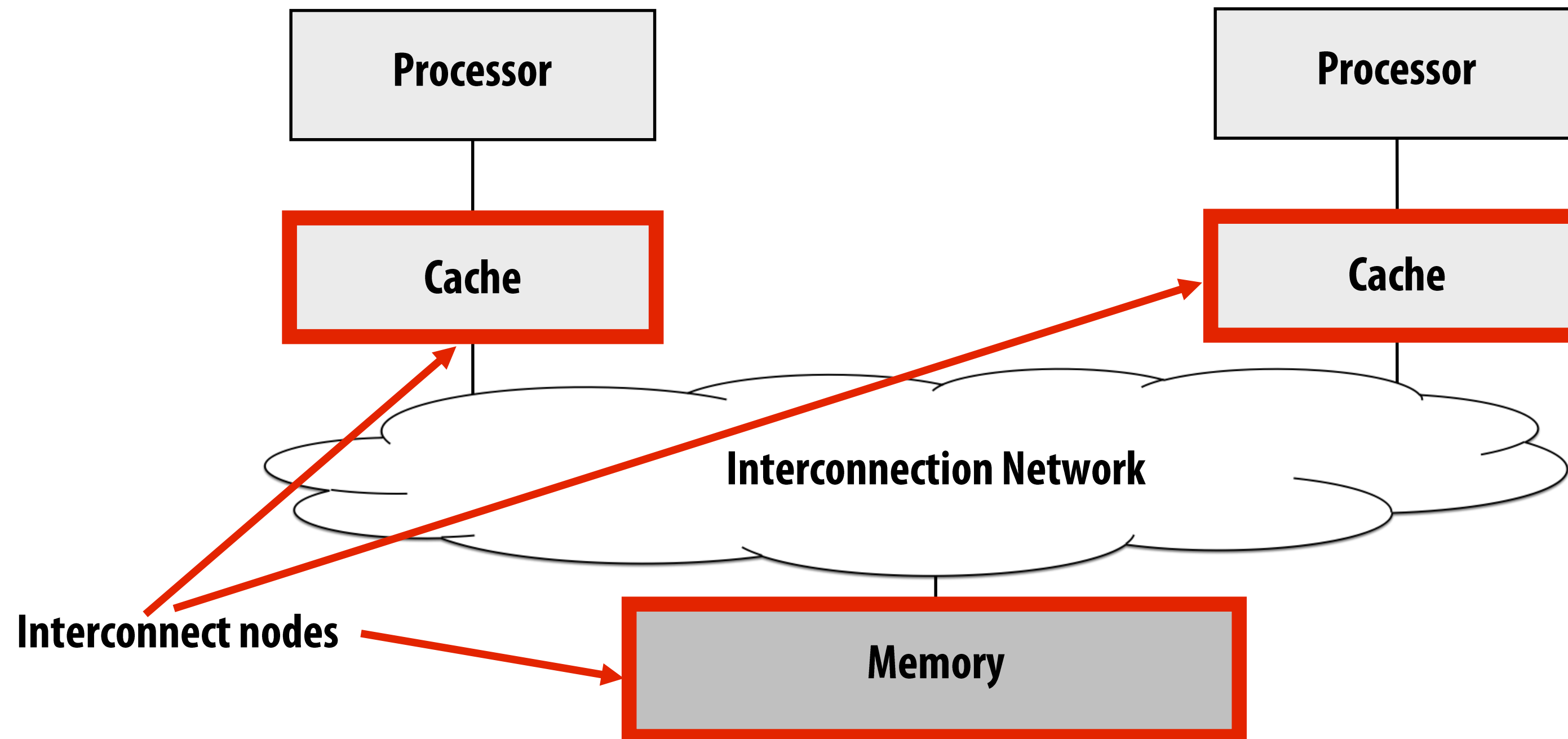


Bus interconnect:

All nodes connected by a shared set of wires



Today: modern interconnect designs



**Today's topics: the basic ideas of building a high-performance interconnection network in a parallel processor.
(think: "a network-on-a-chip")**

What are interconnection networks used for?

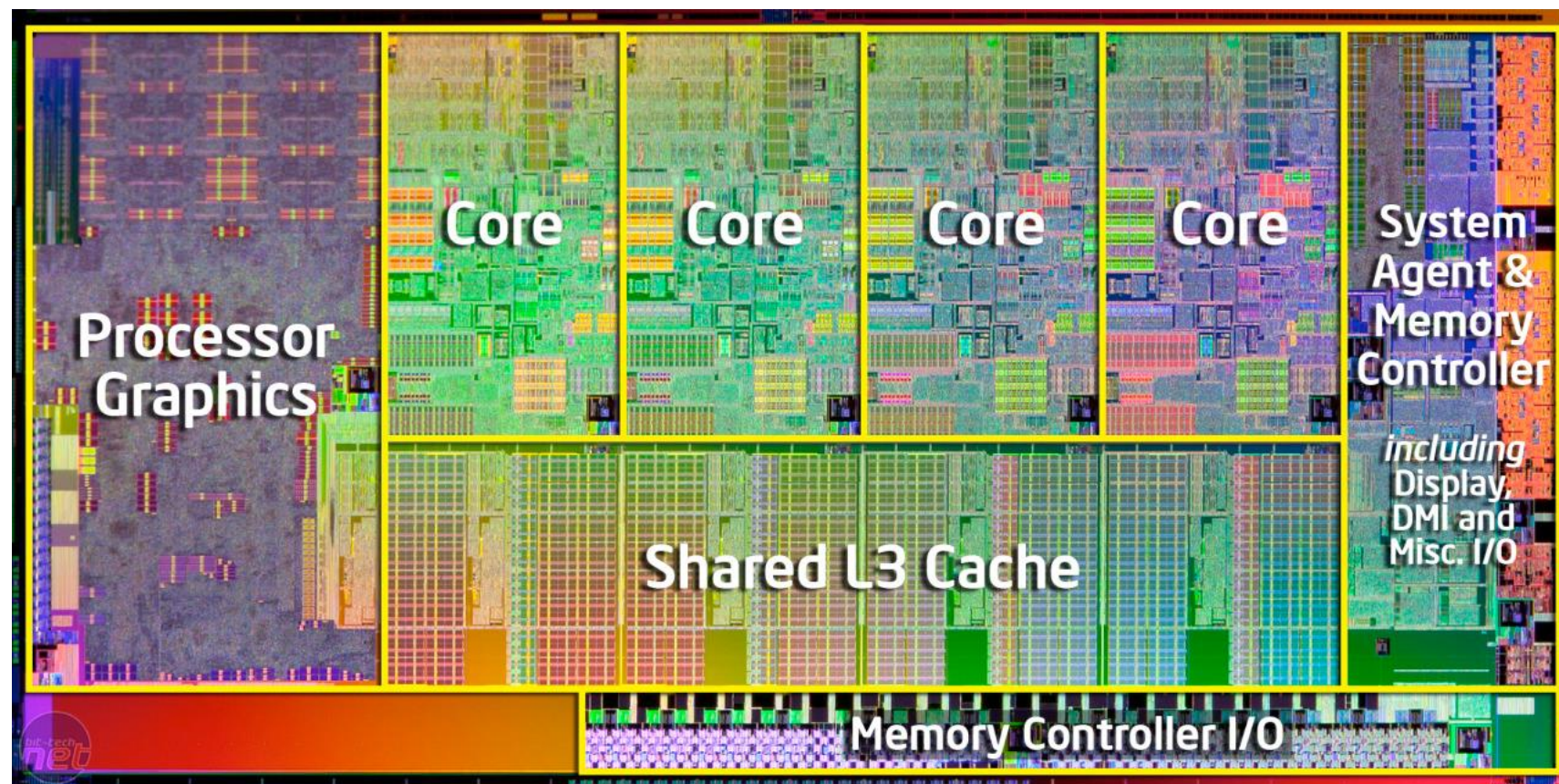
- **To connect:**
 - **Processor cores with other cores**
 - **Processors and memories**
 - **Processor cores and caches**
 - **Caches and caches**
 - **I/O devices**

Why is the design of the interconnection network important?

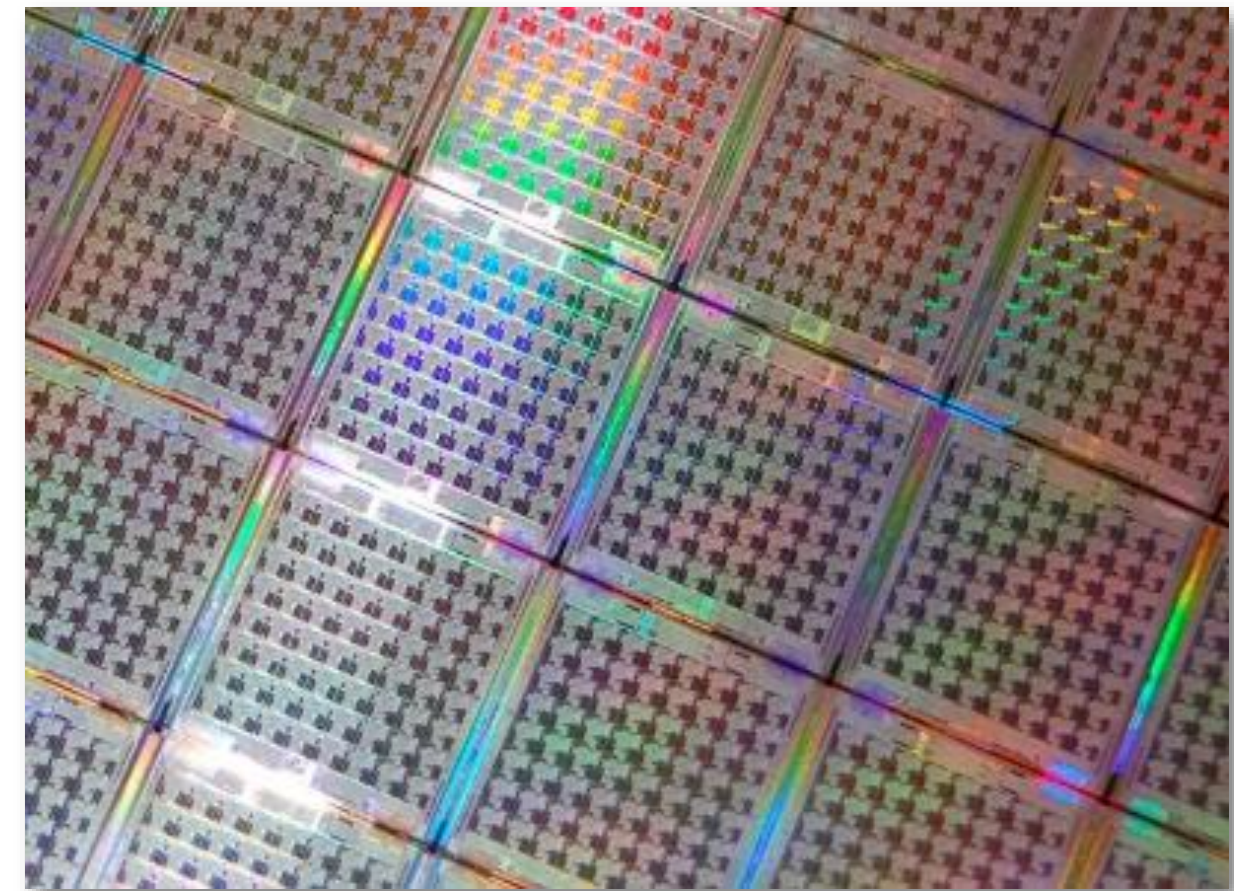
- **System scalability**
 - How large of a system can be built?
 - How easy is it to add more nodes (e.g., cores)
- **System performance and energy efficiency**
 - How fast can cores, caches, memory communicate
 - How long is latency to memory?
 - How much energy is spent on communication?

With increasing core counts...

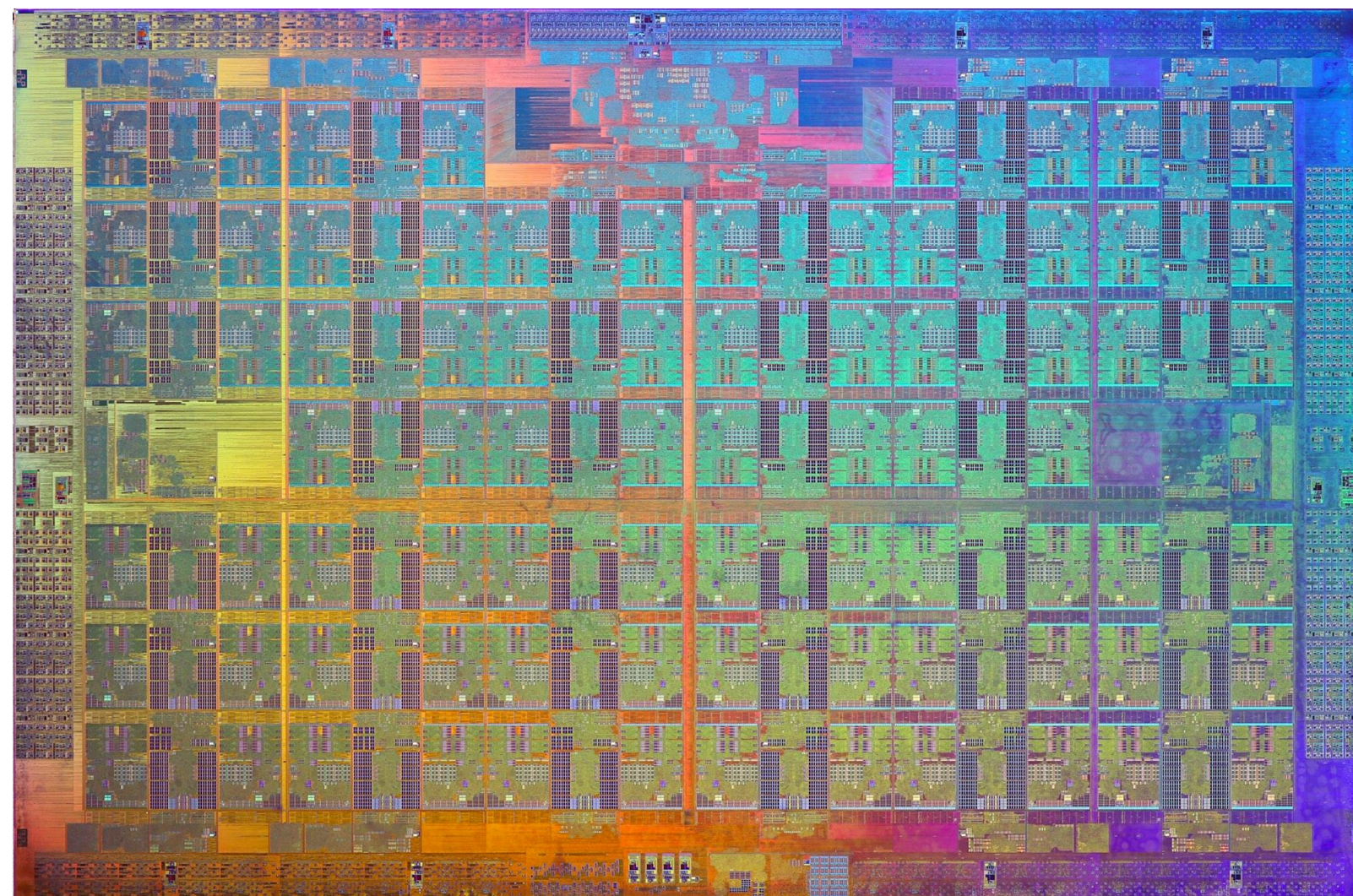
Scalability of on-chip interconnection network becomes increasingly important



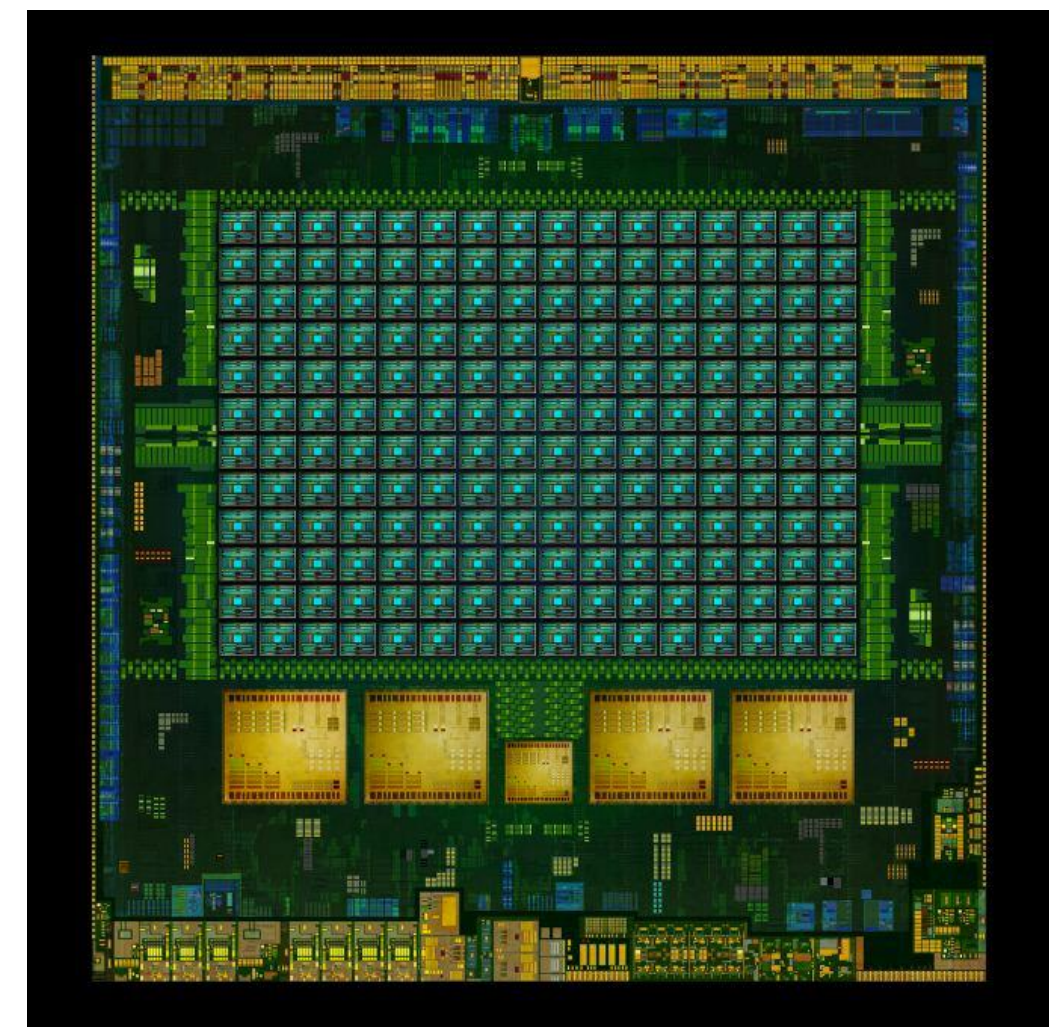
Intel core i7 (4-CPU cores, + GPU)



Tiler GX 64-core chip



Intel Xeon Phi (72-core x86)

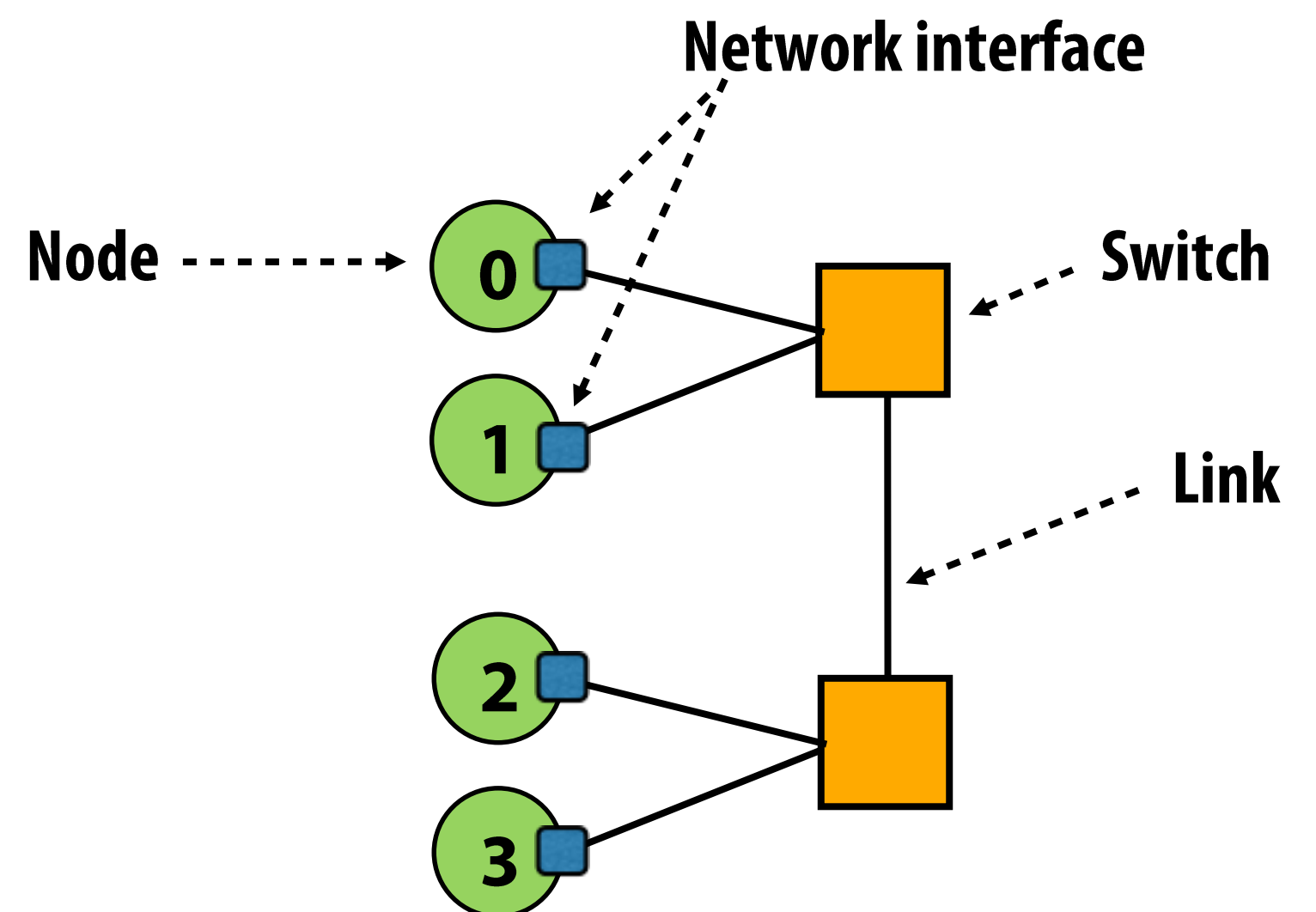


Tegra K1: 4 + 1 ARM cores + GPU cores

Interconnect terminology

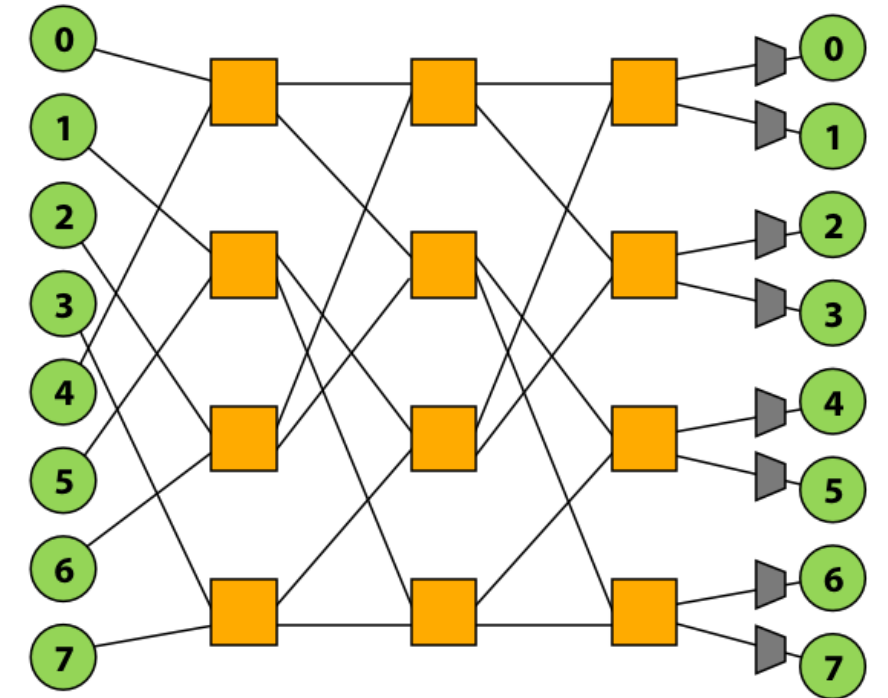
Terminology

- **Network node: a network endpoint connected to a router/switch**
 - Examples: processor caches, the memory controller
- **Network interface:**
 - Connects nodes to the network
- **Switch/router:**
 - Connects a fixed number of input links to a fixed number of output links
- **Link:**
 - A bundle of wires carrying a signal



Design issues

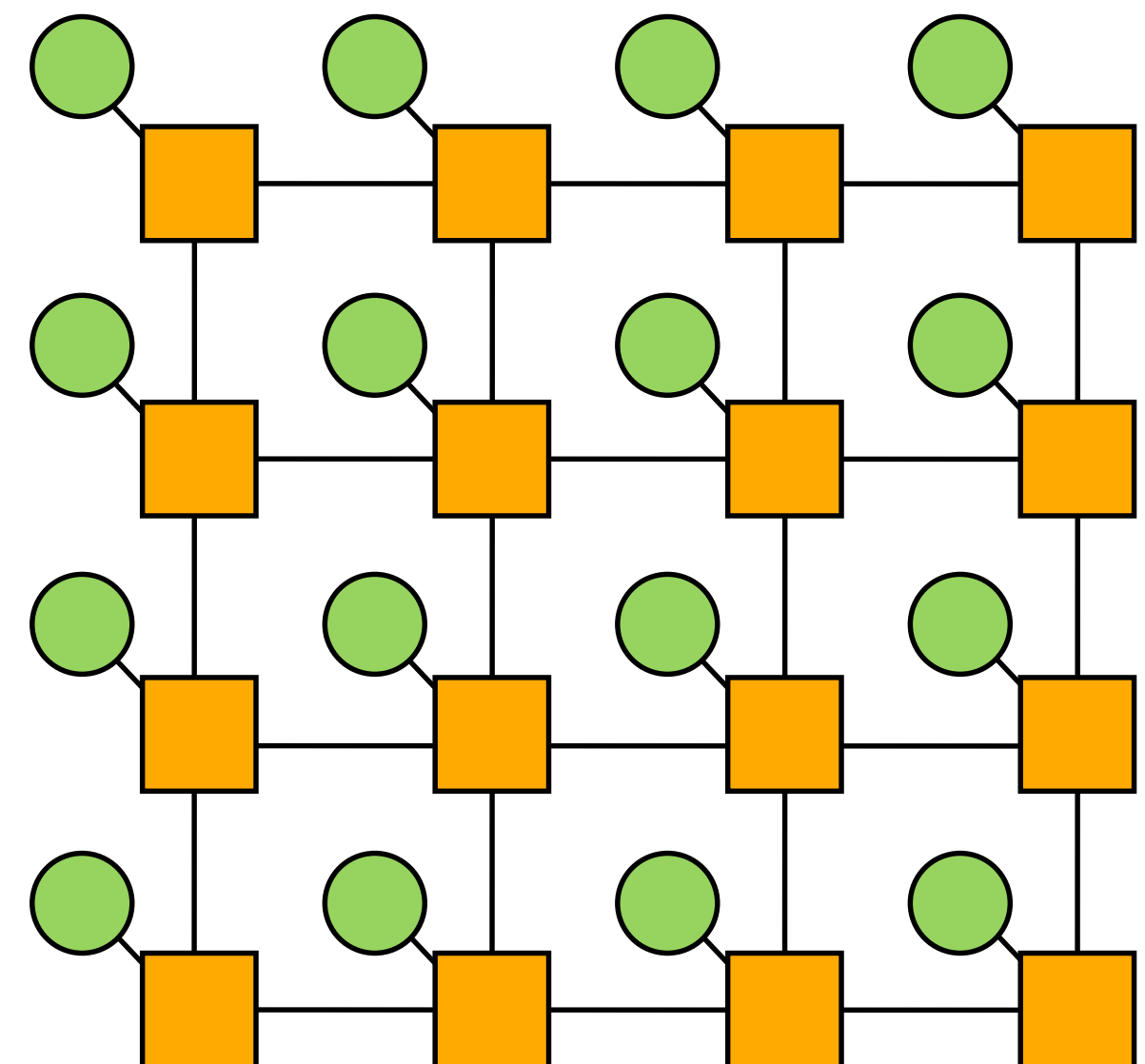
- **Topology: how switches are connected via links**
 - Affects routing, throughput, latency, complexity/cost of implementation
- **Routing: how a message gets from its source to its destination in the network**
 - Can be static (messages take a predetermined path) or adaptive based on load
- **Buffering and flow control**
 - What data is stored in the network? packets, partial packets? etc.
 - How does the network manage buffer space?



Properties of interconnect topology

- **Routing distance**
 - Number of links (“hops”) along a route between two nodes
- **Diameter: the maximum routing distance**
- **Average distance: average routing distance over all valid routes**

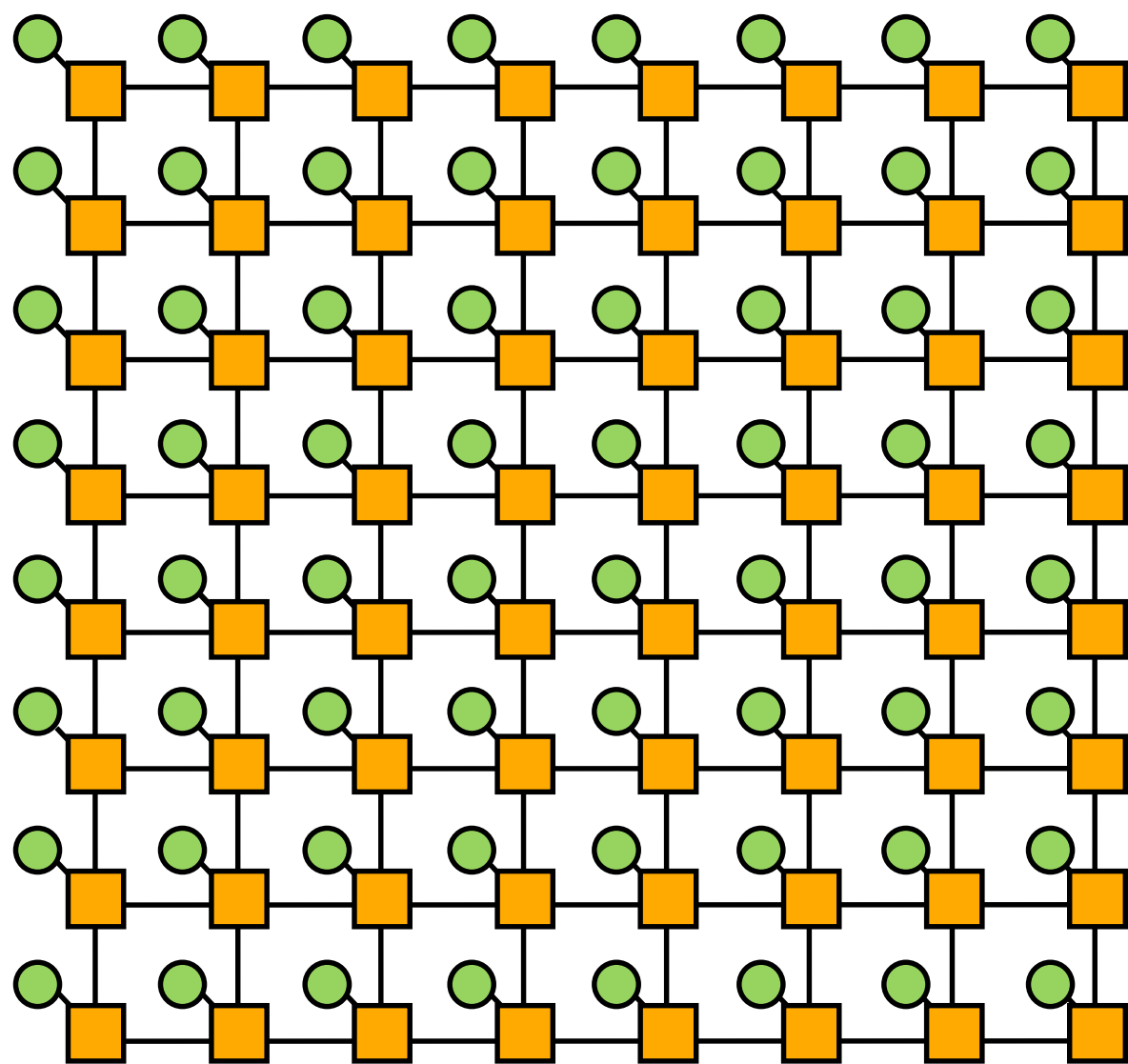
Example:
diameter = 6



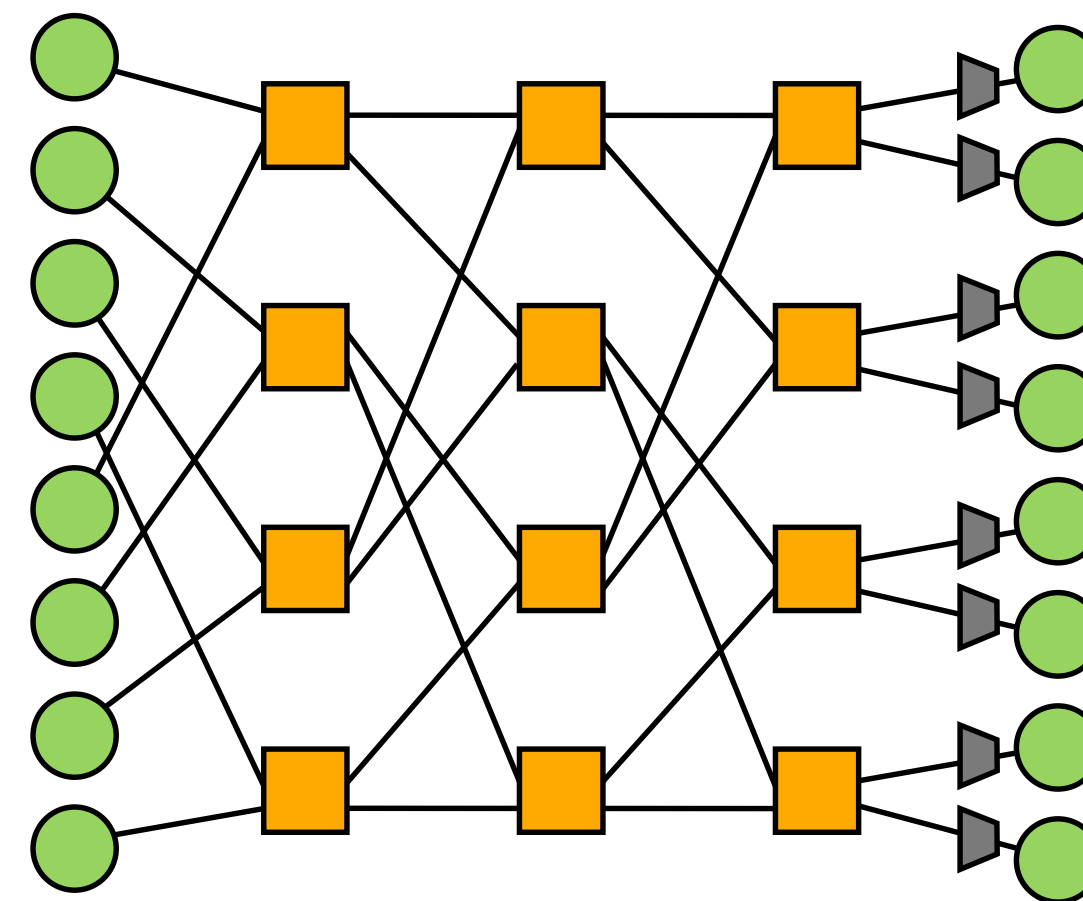
Properties of interconnect topology

■ Direct vs. indirect networks

- **Direct network: endpoints sit “inside” the network**
- **e.g., mesh is direct network: every node is both an endpoint and a switch**



Direct network



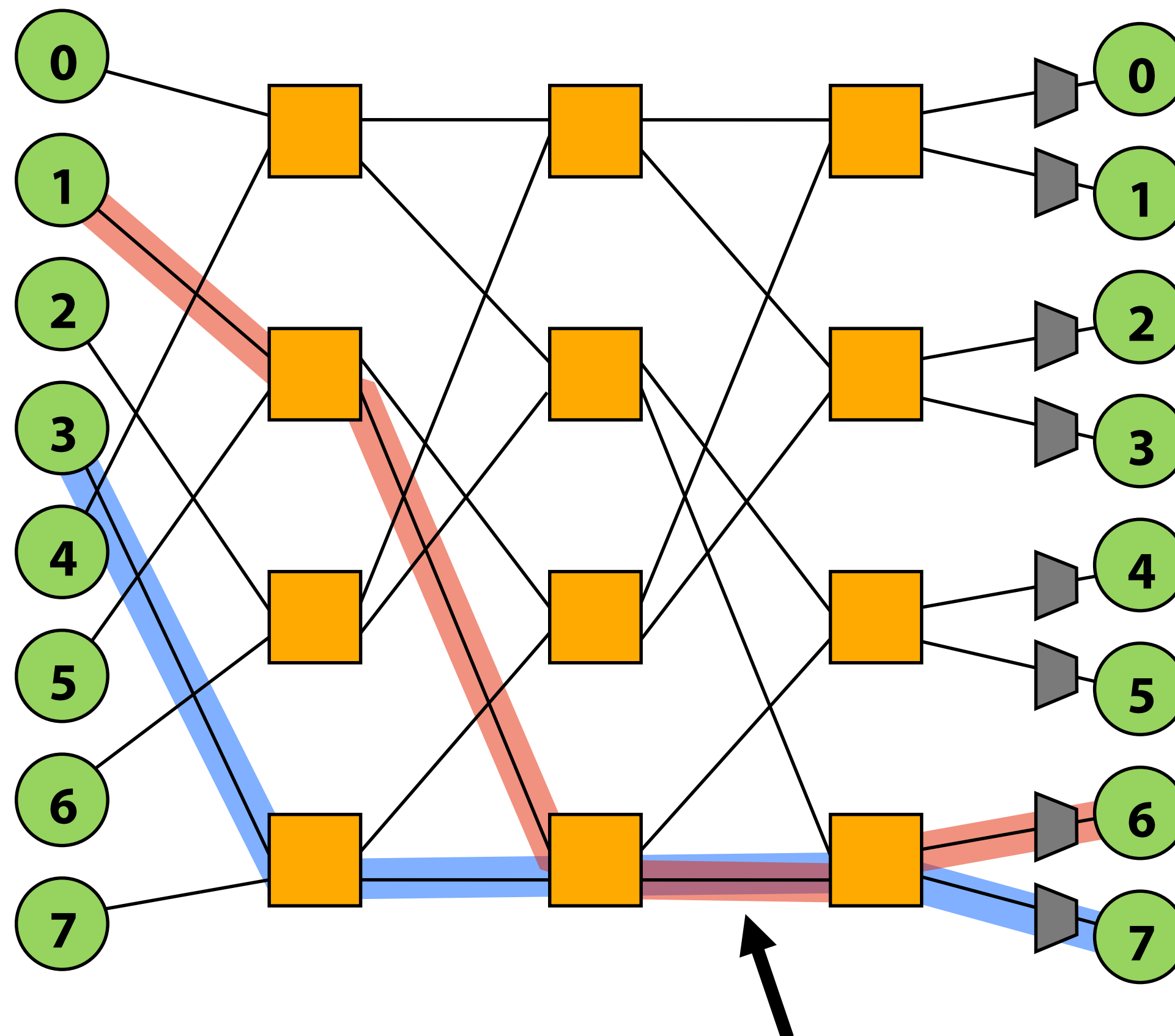
Indirect network

Properties of an interconnect topology

- **Bisection bandwidth:**
 - Common metric of performance for recursive topologies
 - Cut network in half, sum bandwidth of all severed links
 - Warning: can be misleading as it does not account for switch and routing efficiencies
- **Blocking vs. non-blocking:**
 - If connecting any pairing of nodes is possible, network is non-blocking (otherwise, it's blocking)

Example: blocking vs. non-blocking

- Is this network blocking or non-blocking?
 - Consider simultaneous messages from 0-to-1 and 3-to-7.
 - Consider simultaneous messages from 1-to-6 and 3-to-7. **Blocking!!!**

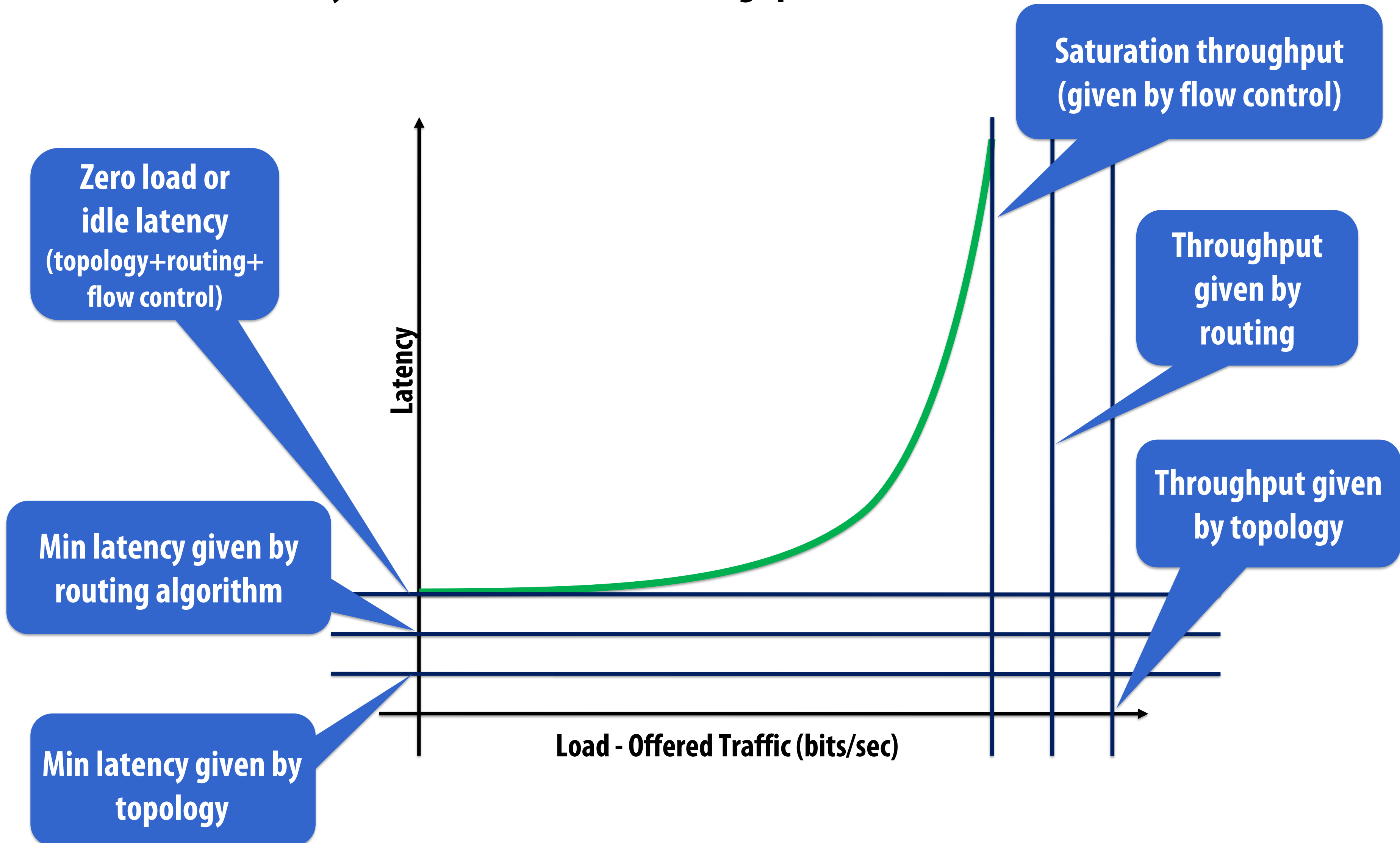


Note: in this network illustration, each node is drawn twice for clarity (at left and at right)

Conflict

Load-latency behavior of network

General rule: latency increases with load (throughput)



Interconnect topologies

Many possible network topologies

Bus

Crossbar

Ring

Tree

Omega

Hypercube

Mesh

Torus

Butterfly

...

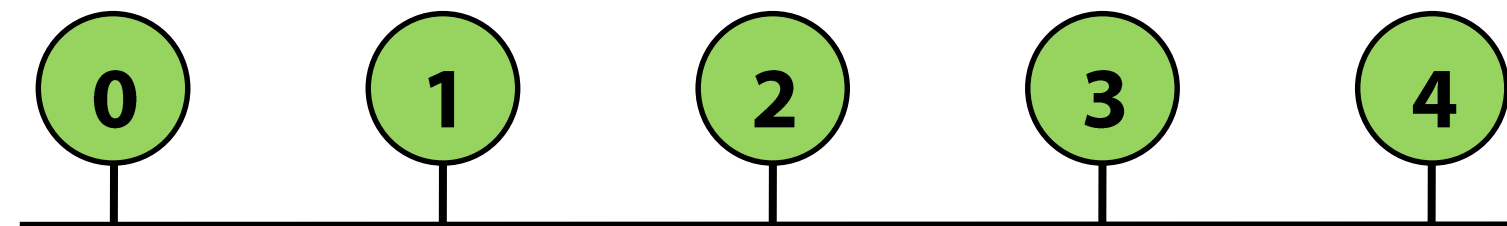
Bus interconnect

- **Good:**

- **Simple design**
- **Cost effective for a small number of nodes**
- **Easy to implement coherence (via snooping)**

- **Bad:**

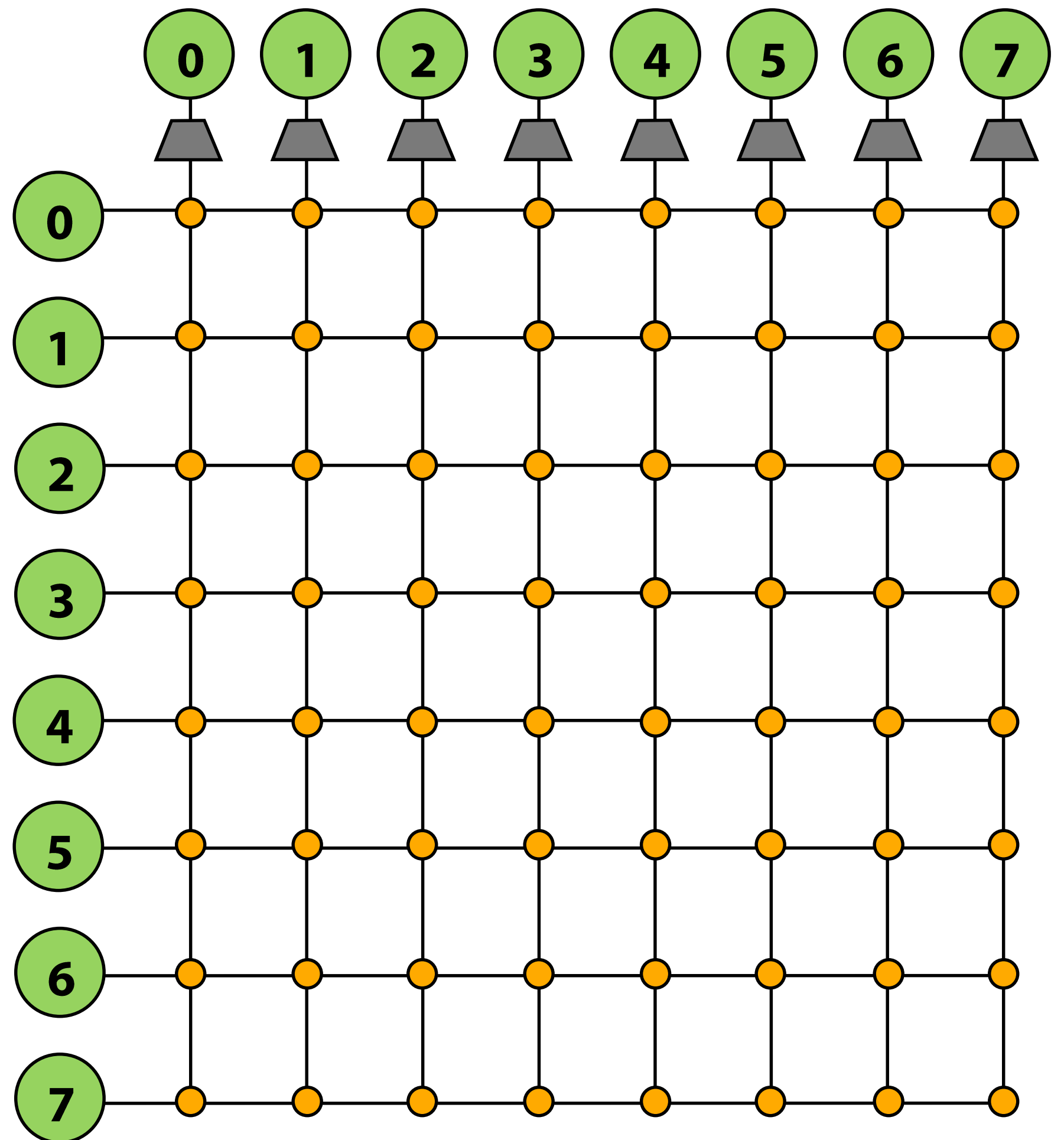
- **Contention: all nodes contend for shared bus**
- **Limited bandwidth: all nodes communicate over same wires (one communication at a time)**
- **High electrical load = low frequency, high power**



Crossbar interconnect

- Every node is connected to every other node (non-blocking, indirect)
- Good:
 - $O(1)$ latency and high bandwidth
- Bad:
 - Not scalable: $O(N^2)$ switches
 - High cost
 - Difficult to arbitrate at scale

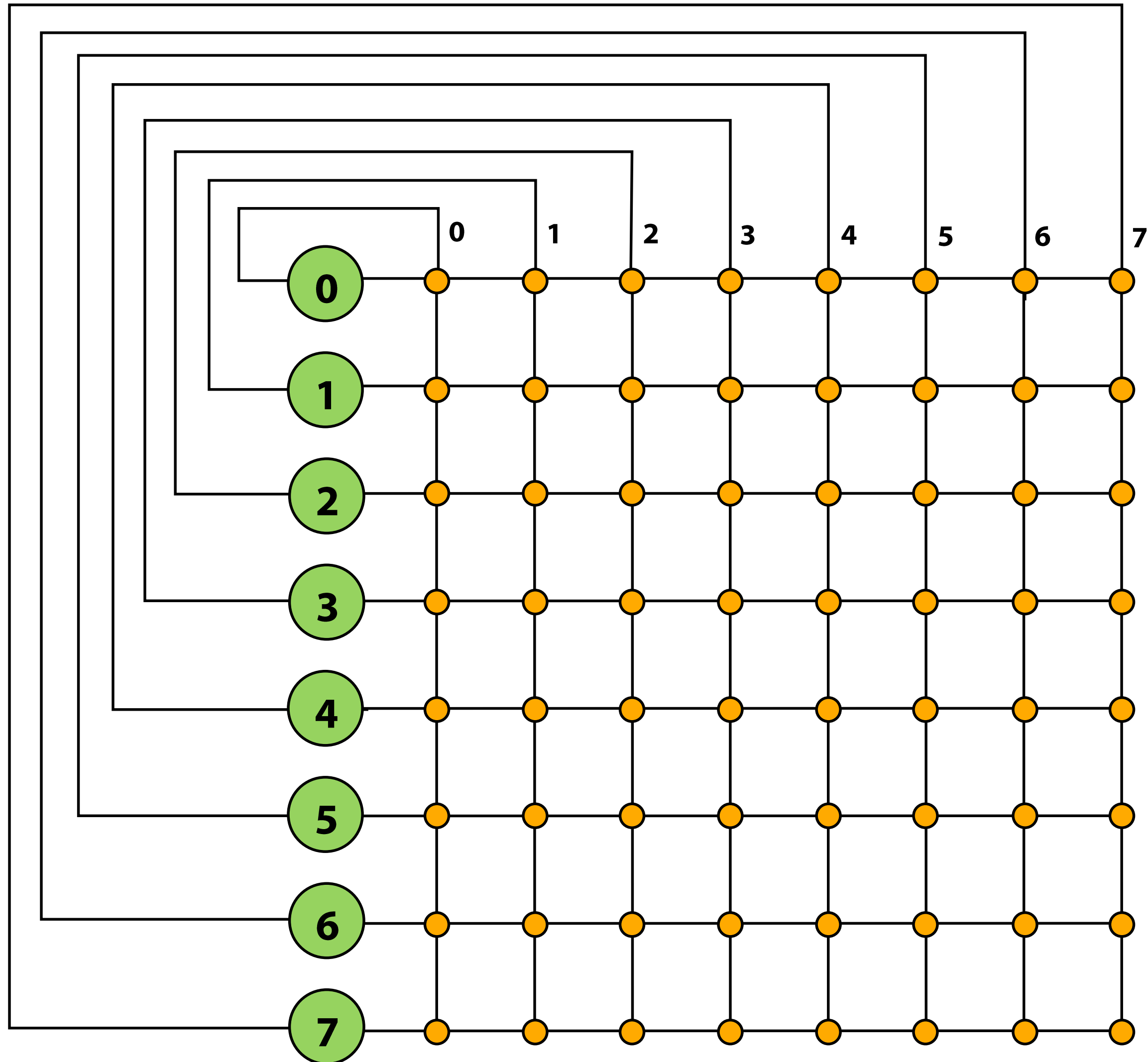
↗
Crossbar scheduling algorithms / efficient hardware implementations are still active research areas.



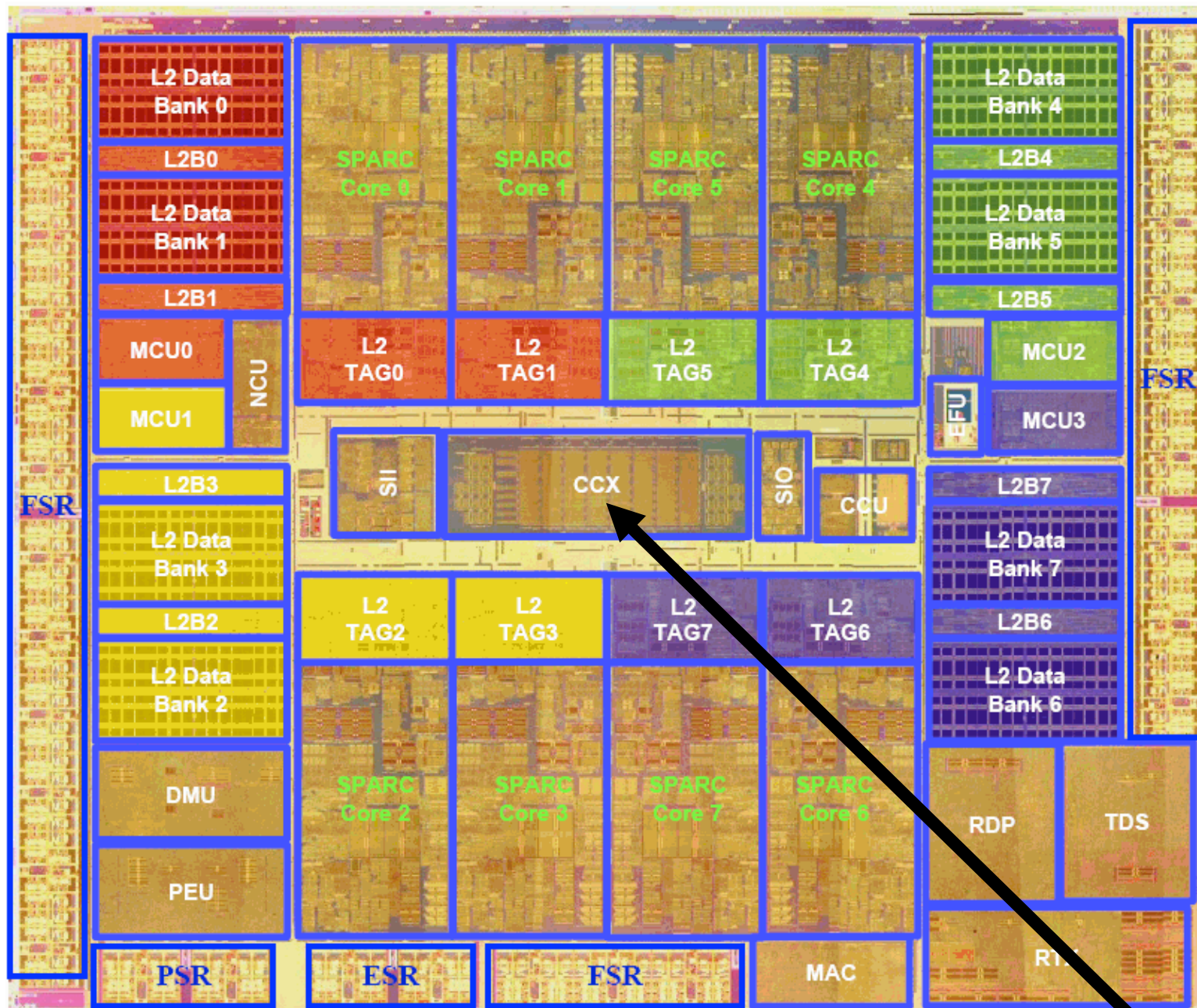
8-node crossbar network (N=8)

Crossbar interconnect

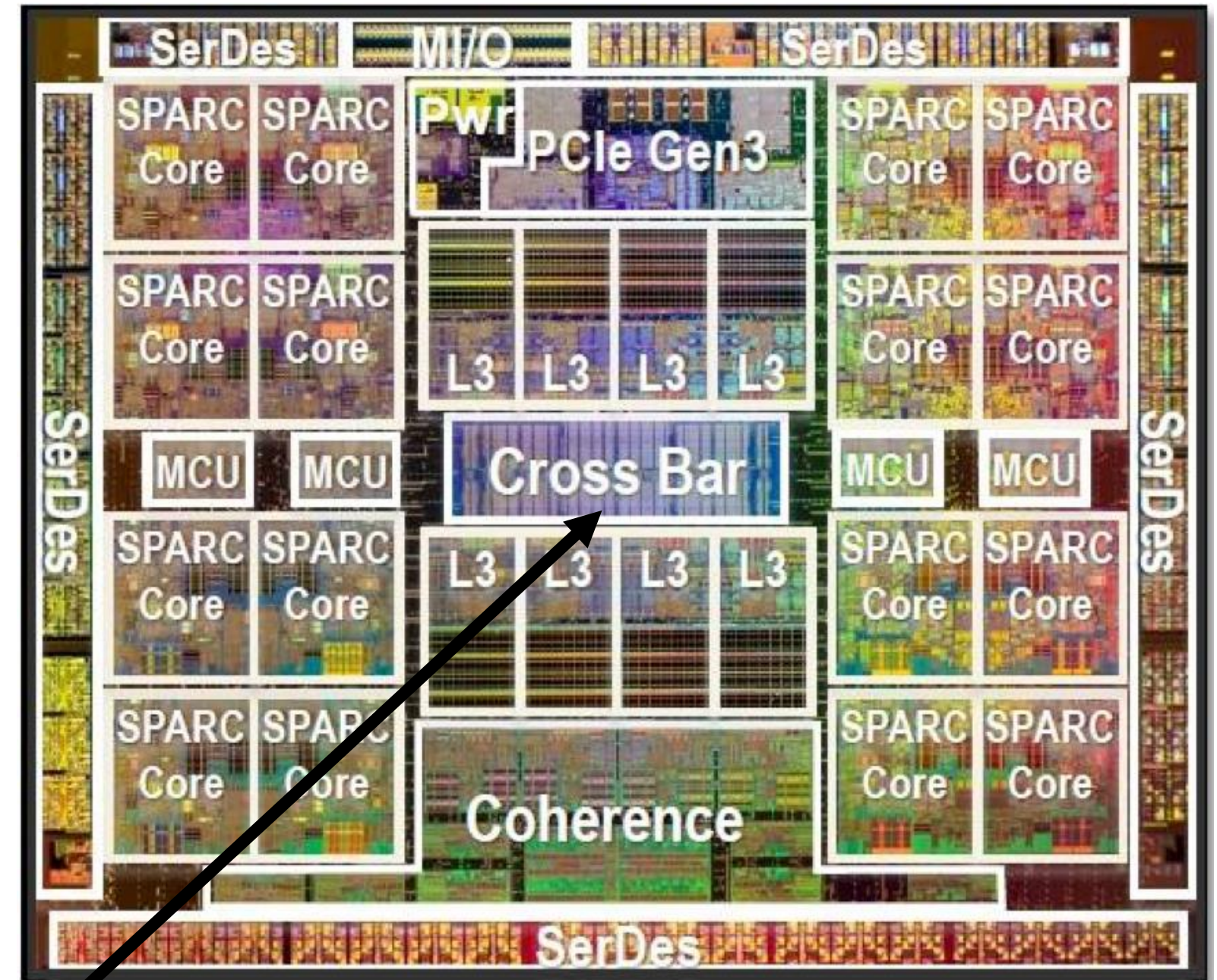
(Here is a more verbose illustration than that on previous slide)



Crossbars were used in recent multi-core processing from Oracle (previously Sun)



Sun SPARC T2 (8 cores, 8 L2 cache banks)

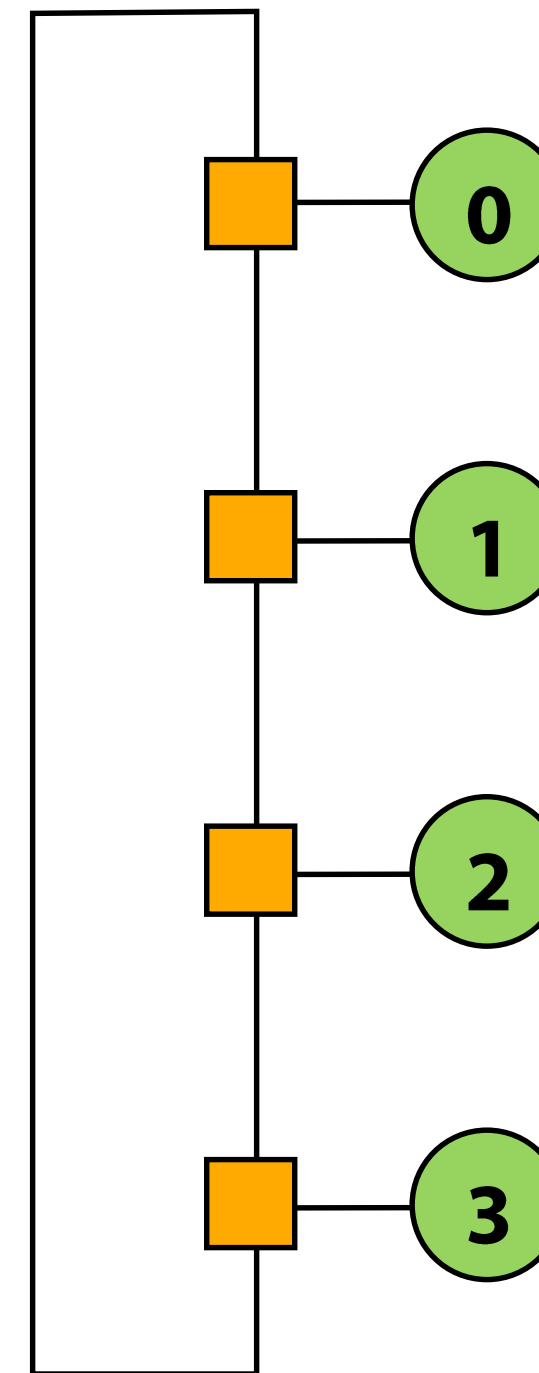


Oracle SPARC T5 (16 cores, 8 L3 cache banks)

Note that crossbar (CCX) occupies about the same chip area as a core

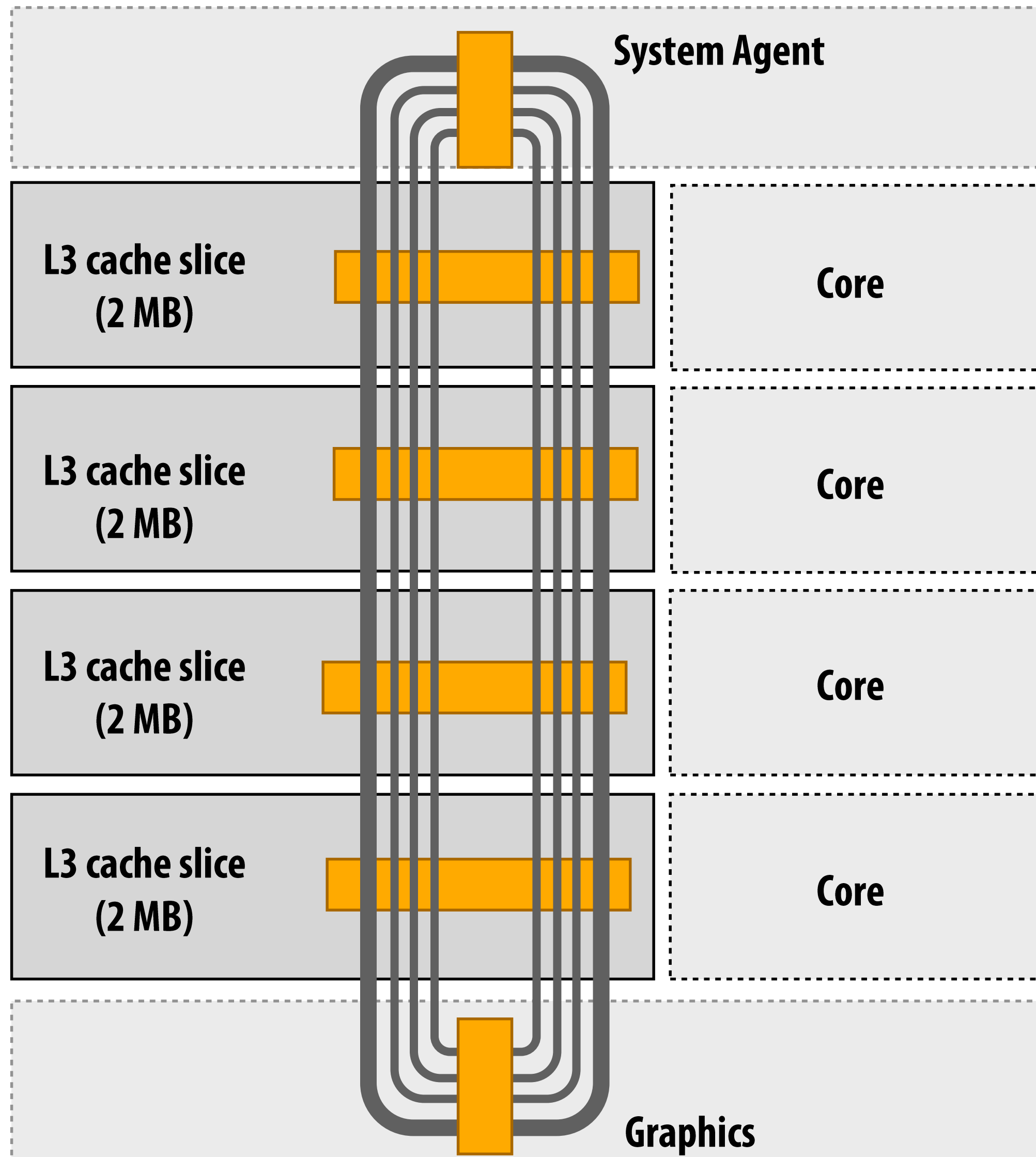
Ring

- **Good:**
 - **Simple**
 - **Cheap: $O(N)$ cost**
- **Bad:**
 - **High latency: $O(N)$**
 - **Bisection bandwidth remains constant as nodes are added (scalability issue)**
- **Used in recent Intel architectures**
 - **Core i7**
- **Also used in IBM CELL Broadband Engine (9 cores)**



Intel's ring interconnect

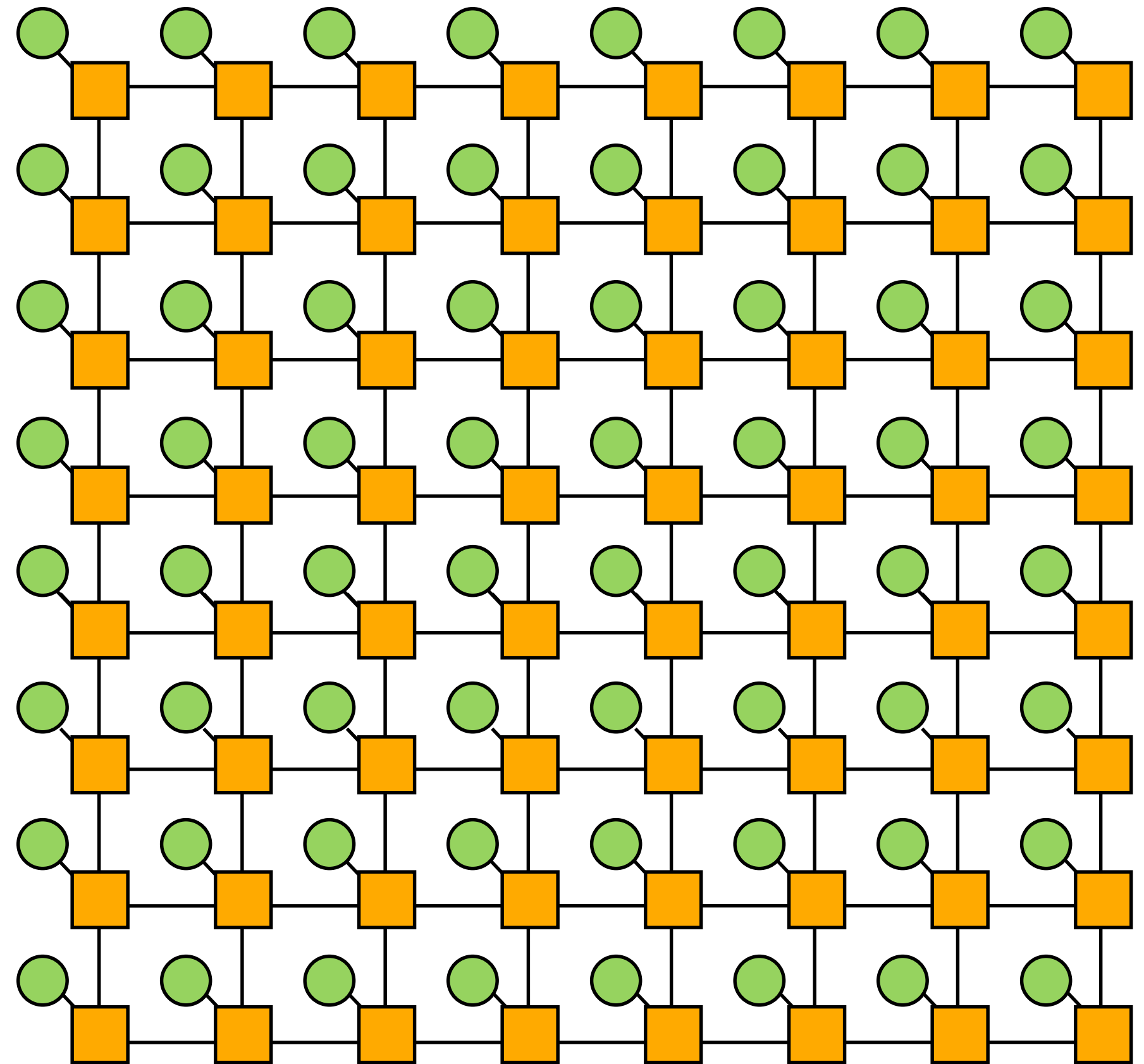
Introduced in Sandy Bridge microarchitecture



- **Four rings**
 - request
 - snoop
 - Ack
 - data (32 bytes)
- **Six interconnect nodes: four “slices” of L3 cache + system agent + graphics**
- **Each bank of L3 connected to ring bus twice**
- **Theoretical peak BW from cores to L3 at 3.4 GHz is approx. 435 GB/sec**
 - When each core is accessing its local slice

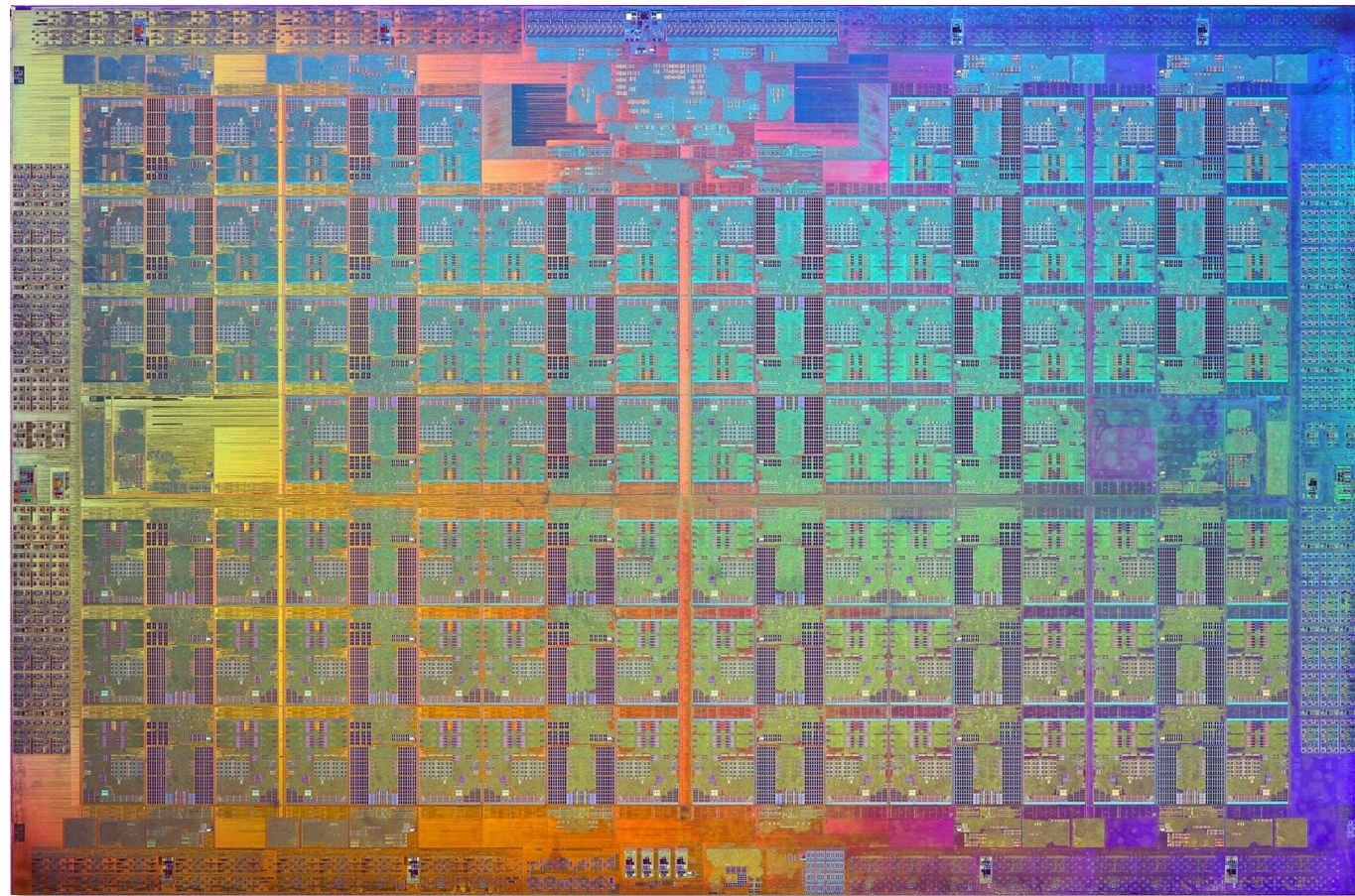
Mesh

- **Direct network**
- **Echoes locality in grid-based applications**
- **$O(N)$ cost**
- **Average latency: $O(\sqrt{N})$**
- **Easy to lay out on chip: fixed-length links**
- **Path diversity: many ways for message to travel from one node to another**
- **Used by:**
 - **Tilera processors**
 - **Prototype Intel chips**

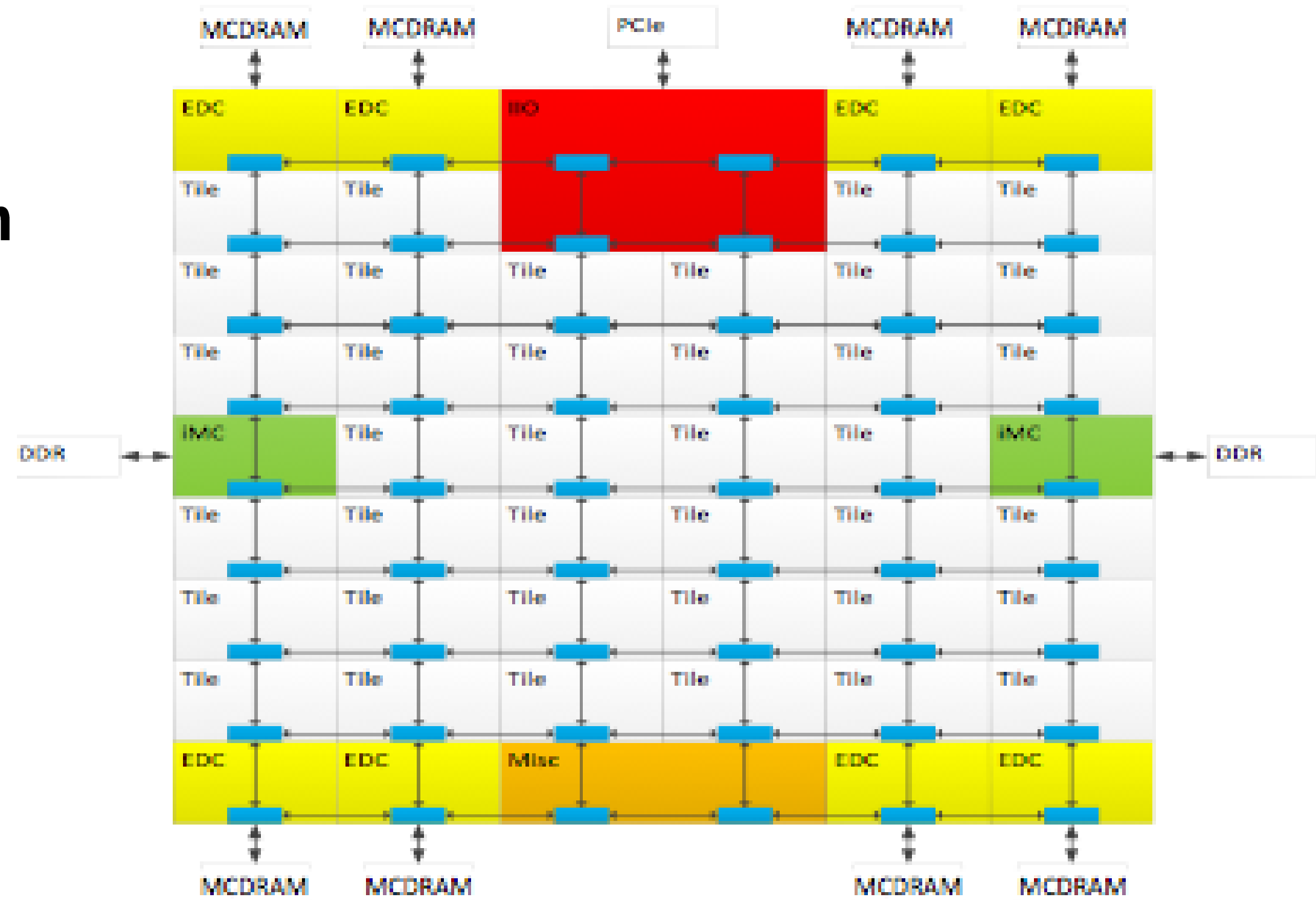


2D Mesh

Xeon Phi (Knights Landing)

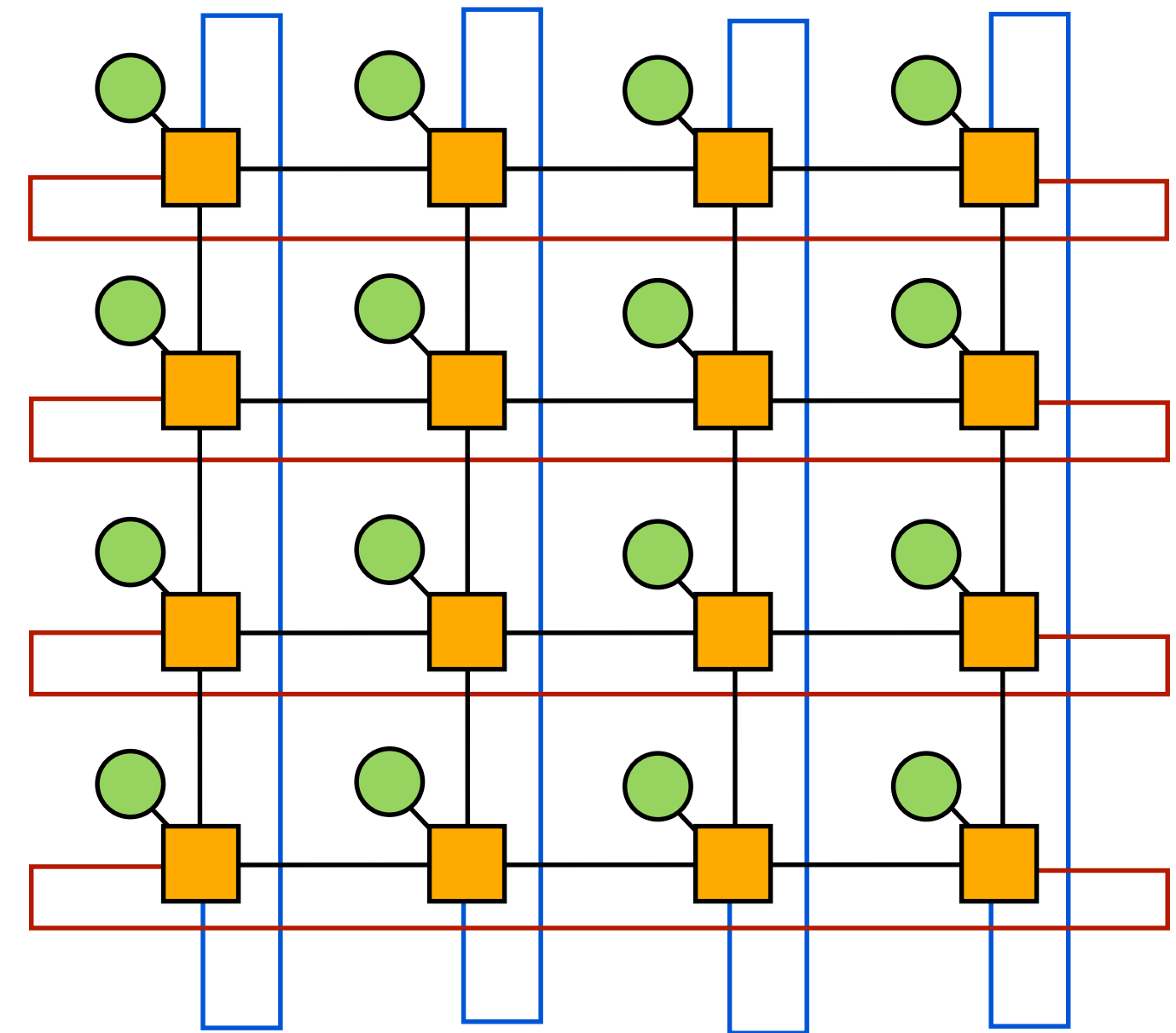


- **72 cores, arranged as 6 x 6 mesh of tiles (2 cores/tile)**
- **YX routing of messages:**
 - **Move in Y**
 - **“Turn”**
 - **Move in X**



Torus

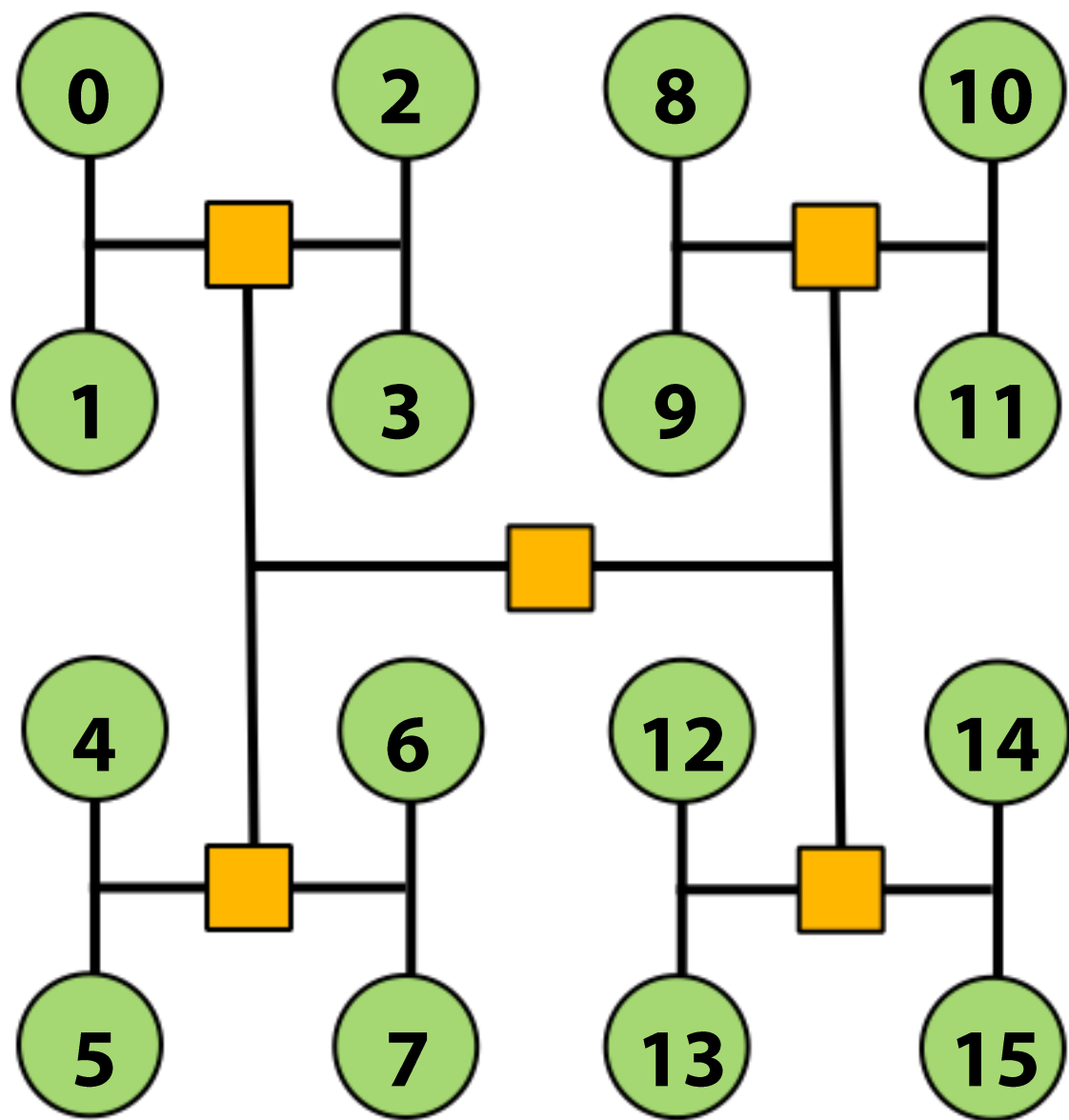
- **Characteristics of mesh topology are different based on whether node is near edge or middle of network (torus topology introduces new links to avoid this problem)**
- **Still $O(N)$ cost, but higher cost than 2D grid**
- **Higher path diversity and bisection BW than mesh**
- **Higher complexity**
 - **Difficult to layout on chip**
 - **Unequal link lengths**



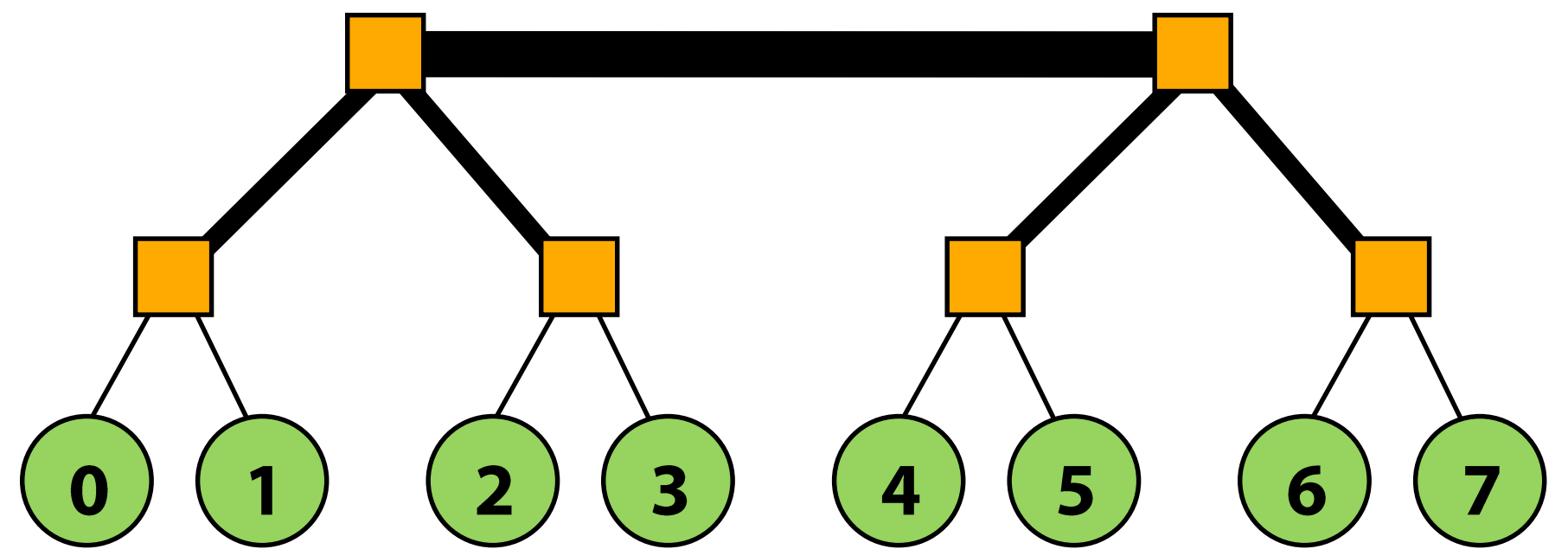
2D Torus

Trees

- Planar, hierarchical topology
- Like mesh/torus, good when traffic has locality
- Latency: $O(\lg N)$
- Use “fat trees” to alleviate root bandwidth problem (higher bandwidth links near root)



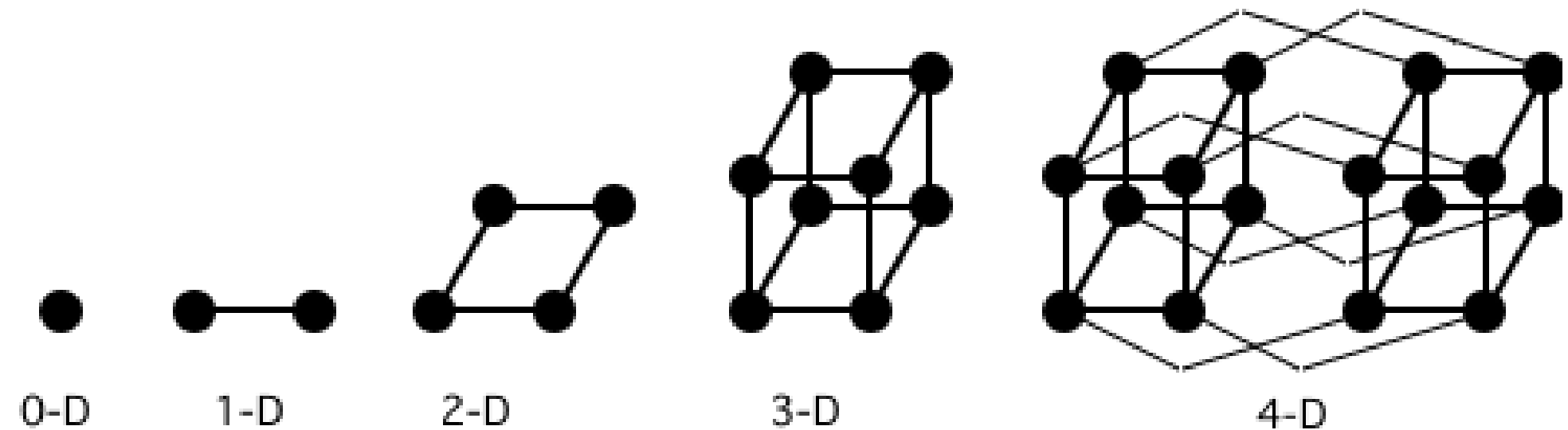
H-Tree



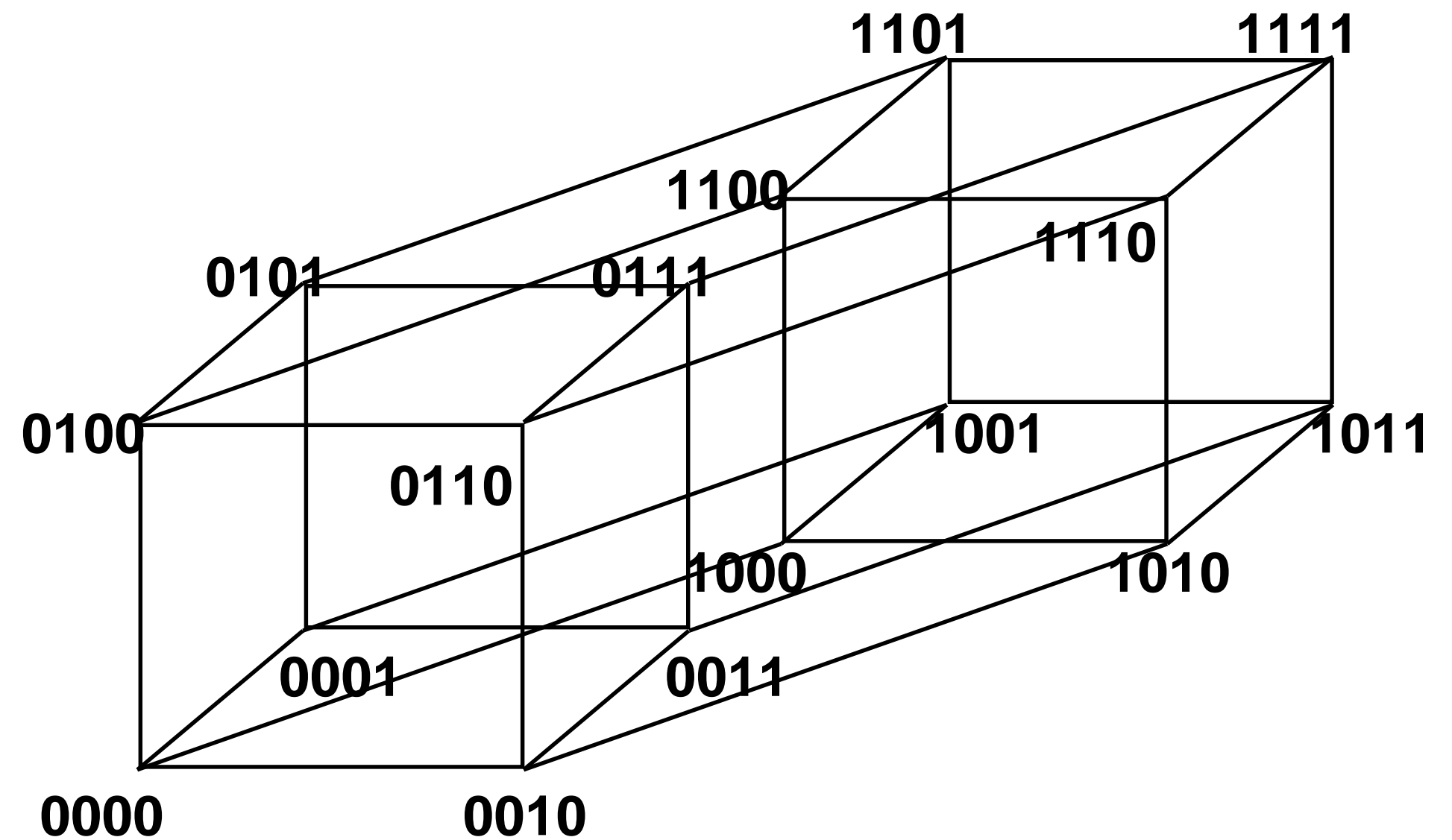
Fat Tree

Hypercube

- **Low latency: $O(\lg N)$**
- **Radix: $O(\lg N)$**
- **Number of links $O(N \lg N)$**

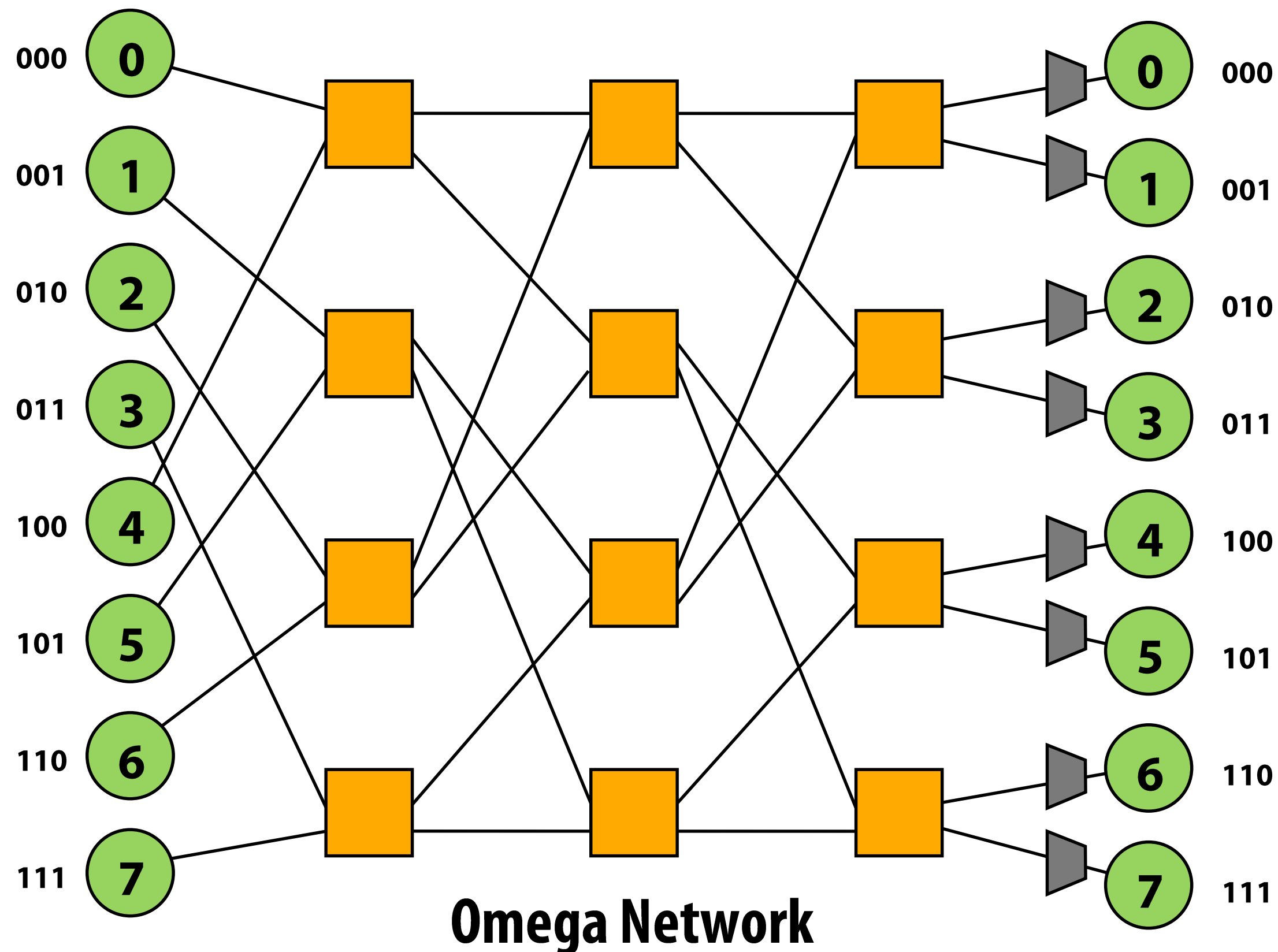


- **6D hypercube used in 64-core Cosmic Cube computer developed at Caltech in the 80s**
- **SGI Origin used a hypercube**

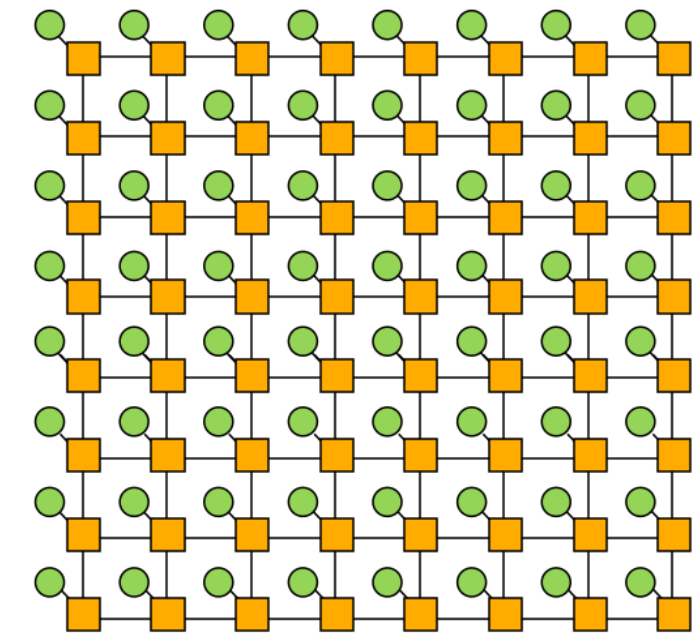
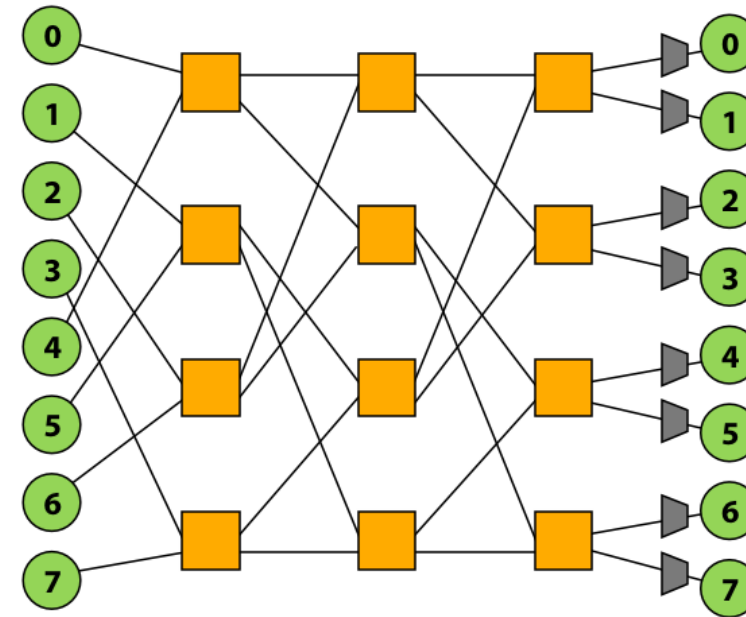
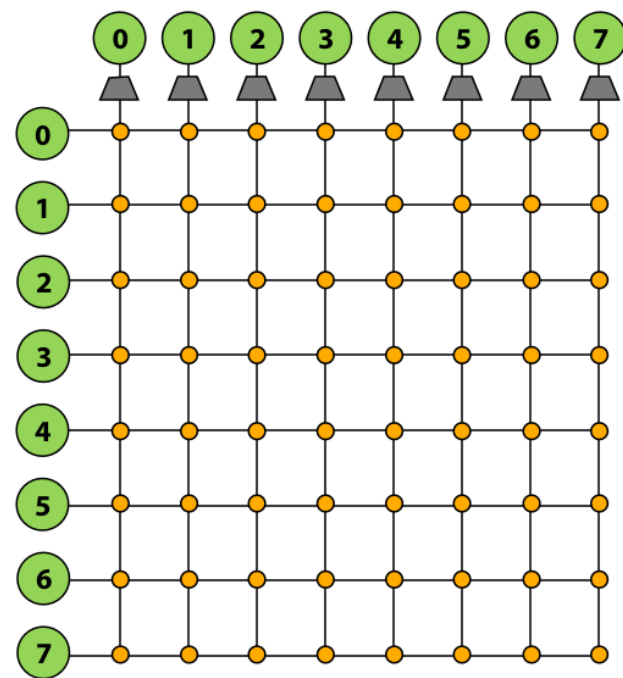


Multi-stage logarithmic

- Indirect network with multiple switches between terminals
- Cost: $O(N \lg N)$
- Latency: $O(\lg N)$
- Many variations: Omega, butterfly, Clos networks, etc...



Review: network topologies



Topology

Crossbar

Multi-stage log.

Mesh

Direct/Indirect

Indirect

Indirect

Direct

**Blocking/
Non-blocking**

Non-blocking

Blocking
(one discussed in class is, others
are not)

Blocking

Cost

$O(N^2)$

$O(N \lg N)$

$O(N)$

Latency

$O(1)$

$O(\lg N)$

$O(\sqrt{N})$
(average)