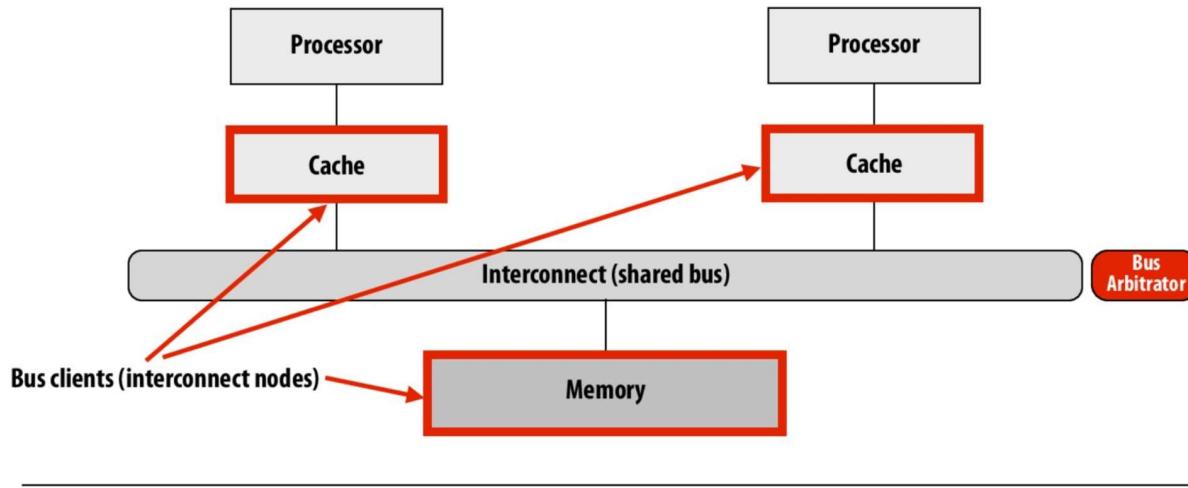


Parallel & Distributed Computing (CS3006)

Lecture 4: On-chip Interconnects

We will be using lectures slides from Professors: **Kayvon Fatahalian** and **Randy Bryant**. These slides are available at: <http://15418.courses.cs.cmu.edu/spring2017/lecture/interconnects> Few slides will be from Prof. Onur Mutlu and Prof. Rana Asif.

Basic system design from previous lectures



Bus interconnect:

Request bus:
e.g., 40 bits cmd + address

All nodes connected by a shared
set of wires

Response bus:
e.g., 256 bits data

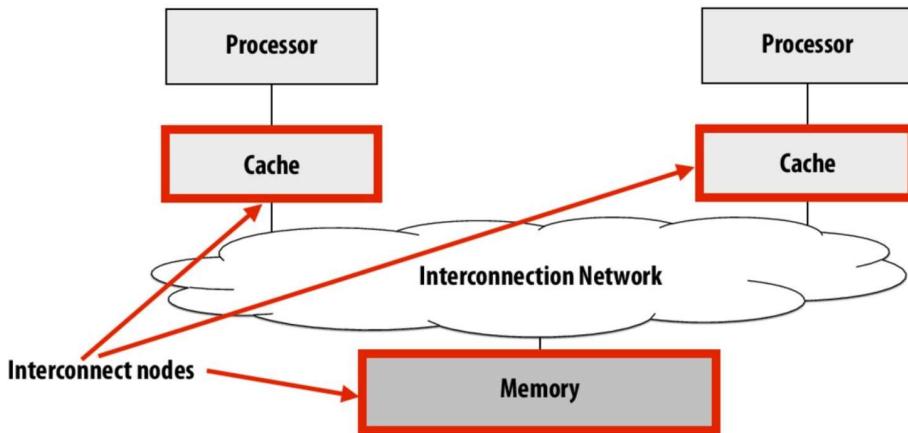
Response tag
3 bits

CMU 15-418/618, Spring 2017

The **use of the bus is serialized** because it is a **shared communication channel** and we do not want two parties talking over each other. Therefore, we need an arbitrator to ensure that only one party is communicating at a time.

Note that processor does not directly talk to the interconnect in this case. Local processor caches are doing so in this example on behalf of their processors.

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")**

CMU 15-418/618, Spring 2017

All of your knowledge from your computer networking class will come into play. But it will be a network on the 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

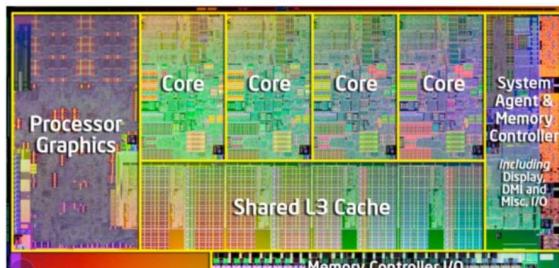
- How fast can cores, caches, memory communicate
- How long is latency to memory?
- How much energy is spent on communication?

CMU 15-418/618, Spring 2017

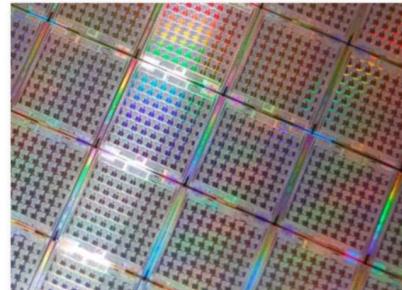
We should think about Amdahl's law in this context as well. What if we are making cores very efficient but totally ignoring the interconnects? How doing so will impact the performance?

With increasing core counts...

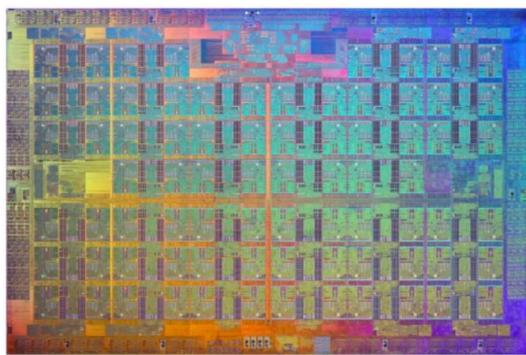
Scalability of on-chip interconnection network becomes increasingly important



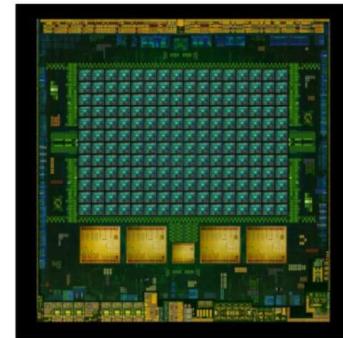
Intel core i7 (4-CPU cores, + GPU)



Tilera GX 64-core chip



Intel Xeon Phi (72-core x86)



Tegra K1: 4 + 1 ARM cores + GPU cores

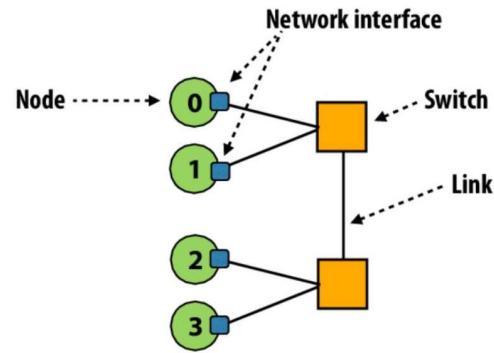
CMU 15-418/618, Spring 2017

Basically, having many processors won't be very helpful if they all have to communicate serially on the same bus. That's why we have different interconnection networks to remove the serial bottleneck.

Some 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



CMU 15-418/618, Spring 2017

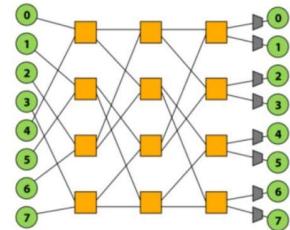
What's the difference between switch and a router?

If processor 2 needs to send something to processor 3, will send message to switch, switch looks at destination, and switch makes routing decision.

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**

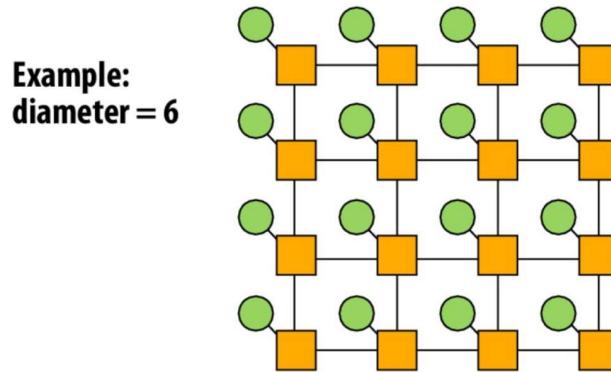
- What data is stored in the network? packets, partial packets? etc.
 - How does the network manage buffer space?

CMU 15-418/618, Spring 2017

What's the difference between routing and forwarding? Routing is the process by which switches understand the current topology of the network. The end product of one routing run is a forwarding table. A forwarding table simply has rules like: if a packet for destination x comes to your input port A, and it is destined to D, emit it at your port B etc.

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**



CMU 15-418/618, Spring 2017

Question: Is a hop a jump between 2 switches/routers as opposed to a jump between a switch and a node? If its the latter then the diameter should be 8 right?

Answer:

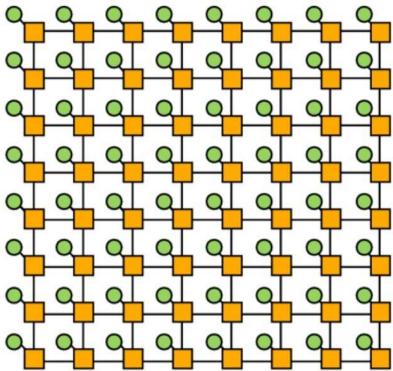
Since for any network the node-switch links are necessary, it is not counted towards bandwidth.

Here nodes are acting as both endpoints and switches. Thus, we are not considering link between endpoint and switch towards diameter.

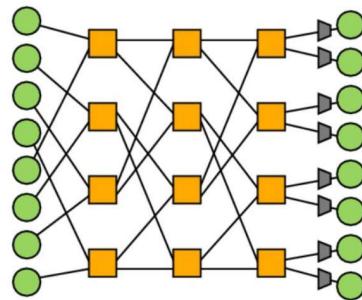
Properties of interconnect topology

■ 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

CMU 15-418/618, Spring 2017

Direct networks are at times called static networks as well.

Indirect networks are at times called dynamic networks as well.

Question: What will be the diameter for direct and indirect networks above? (Be careful with your assumptions. Better write them down. Few people argue that for indirect networks, we should count the links connecting nodes to switches because those are external to the processor.)

Properties of an interconnect topology

■ Bisection bandwidth:

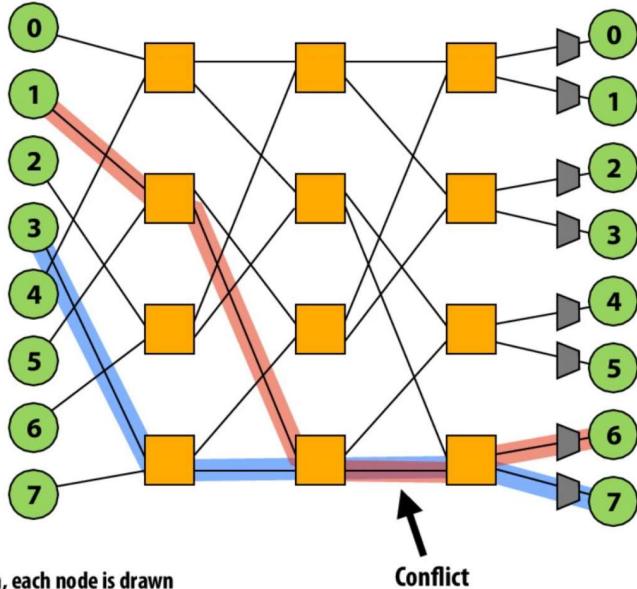
- Common metric of performance for recursive topologies
- Cut network in half, sum bandwidth of all severed links
- Warning: can be misleading as it does not account for switch and routing efficiencies

■ Blocking vs. non-blocking:

- If connecting any pairing of idle 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)

CMU 15-418/618, Spring 2017

Interconnect Topologies:

Many possible network topologies

Bus

Crossbar

Ring

Tree

Omega

Hypercube

Mesh

Torus

Butterfly

...

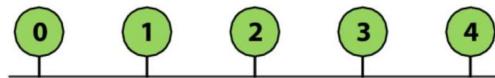
Bus interconnect

■ 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



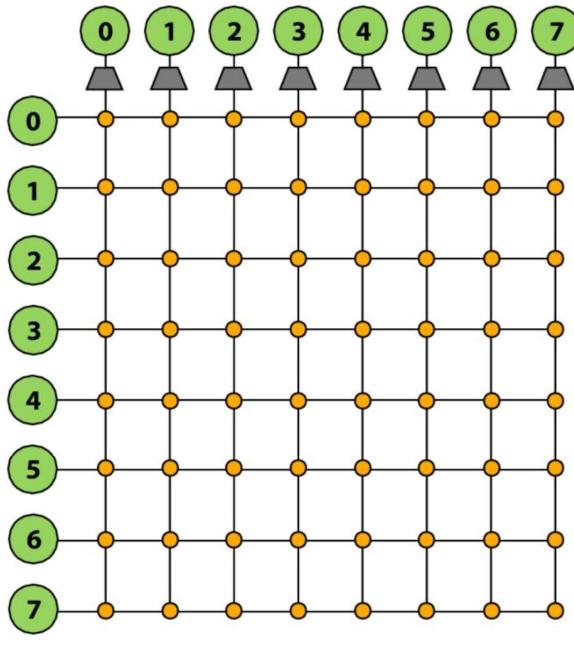
CMU 15-418/618, Spring 2017

When we say cost, we need to be clear cost in what terms? Is it cost as in dollars or cost in some other metric like amount of hardware (that can indirectly translate to the dollar cost.)

Above no switch needed. So cost effective in the sense that we don't need switch. (Remember this all is on the chip die, where saving small silicon area is usually a huge benefit because something else, something more useful could use that space.)

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



8-node crossbar network ($N=8$)

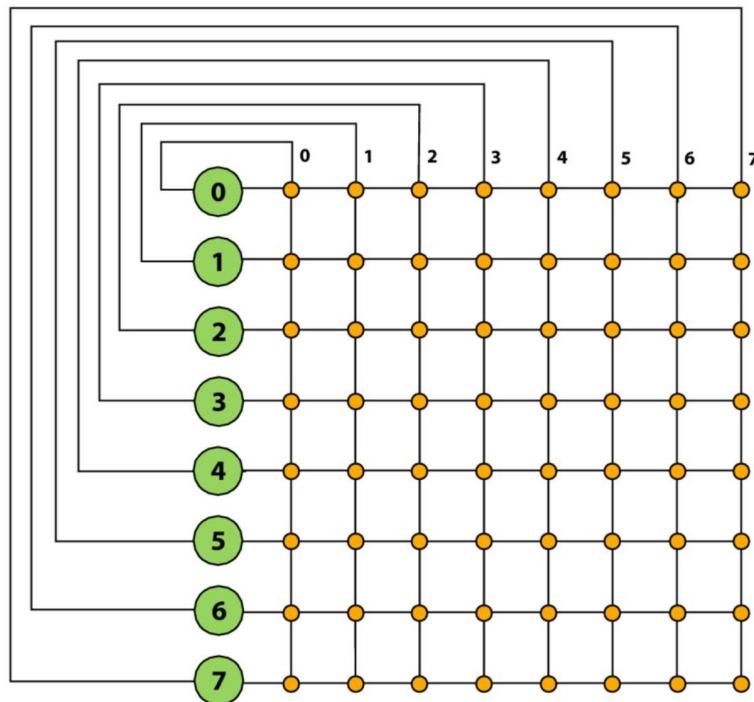
Note: in this network illustration, each node is drawn twice for clarity (at left and at top)

CMU 15-418/618, Spring 2017

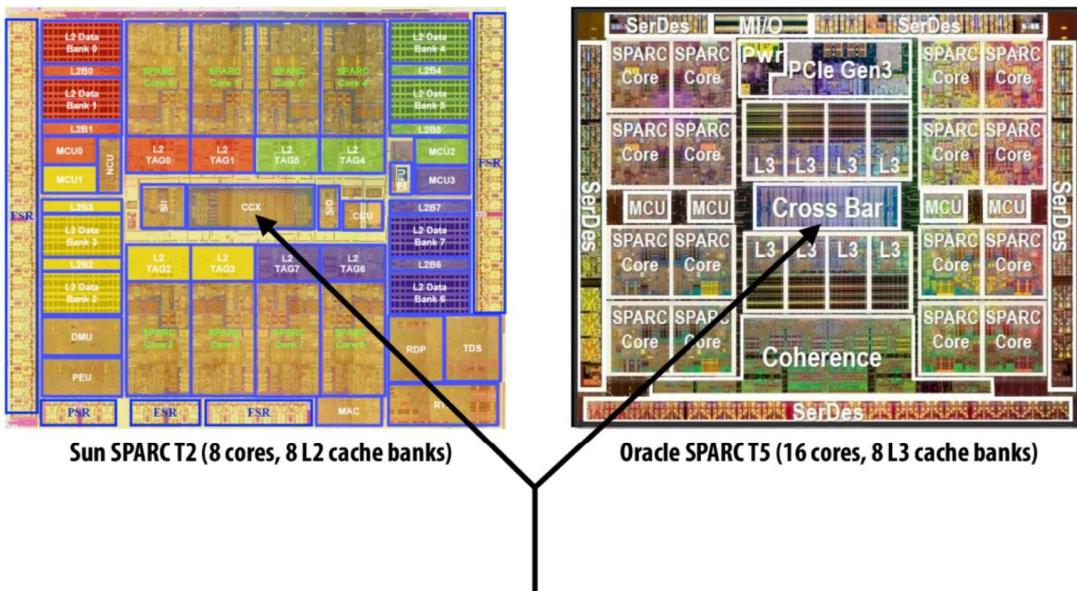
Question: Is crossbar a direct network or an indirect network? (Again write your assumption and reasons clearly.) Think direct networks as in: point-to-point links.

Crossbar interconnect

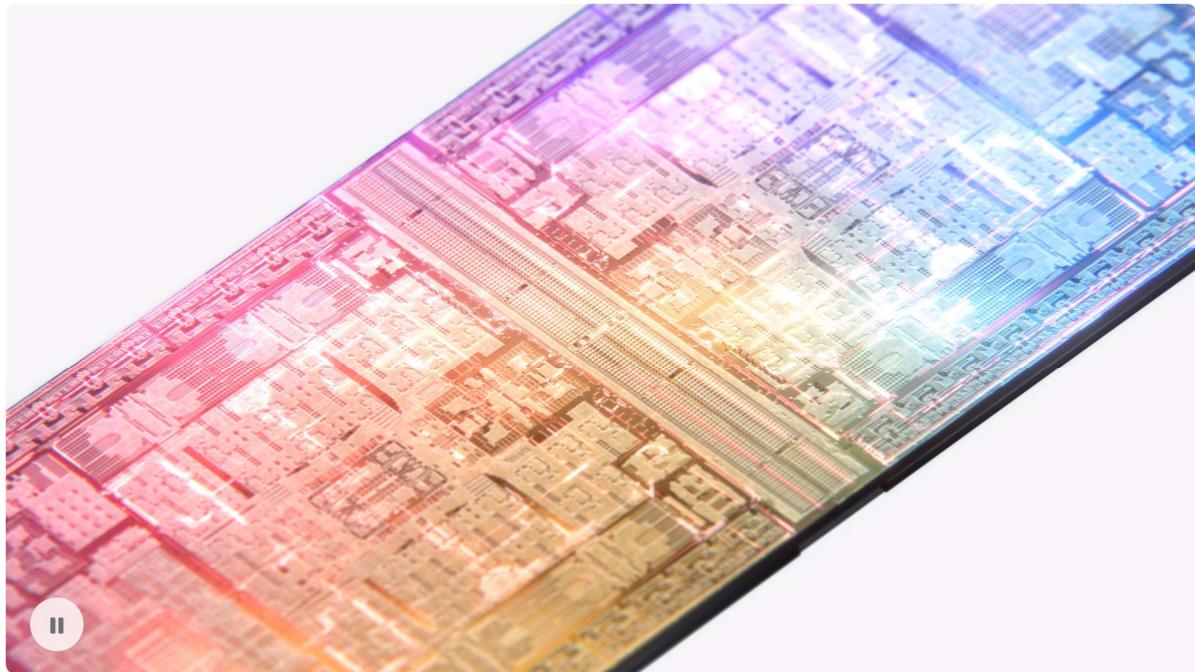
(Here is a more verbose illustration than that on previous slide)



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



CMU 15-418/618, Spring 2017



Apple's UltraFusion packaging architecture uses a silicon interposer that connects the die of two M2 Max chips to create M2 Ultra.



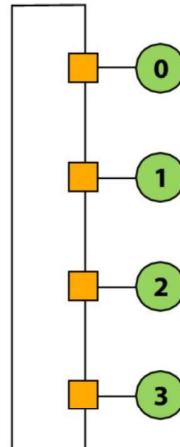
"M2 Ultra is built from two M2 Max dies connected through UltraFusion, Apple's industry-leading, custom-built packaging technology.

UltraFusion uses a silicon interposer that connects the dies with more than **10,000 signals**, providing over **2.5TB/s of low-latency** interprocessor bandwidth." [Taken from:

[https://www.apple.com/newsroom/2023/06/apple-introduces-m2-ultra/#:~:text=Apple's%20UltraFusion%20packaging%20architecture%20uses,chip%20to%20create%20M2%20Ultra.\]](https://www.apple.com/newsroom/2023/06/apple-introduces-m2-ultra/#:~:text=Apple's%20UltraFusion%20packaging%20architecture%20uses,chip%20to%20create%20M2%20Ultra.)

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)

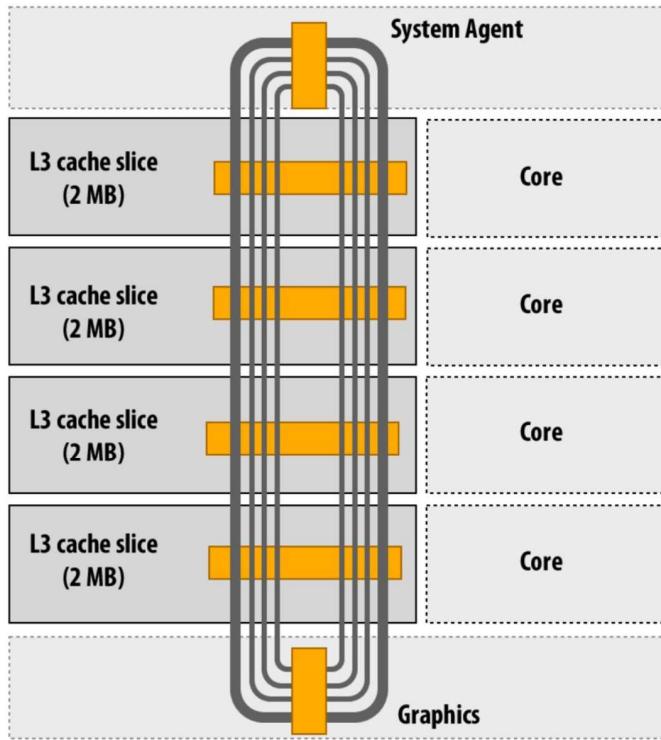


CMU 15-418/618, Spring 2017

Question: How is it different from Bus?

Intel's ring interconnect

Introduced in Sandy Bridge microarchitecture

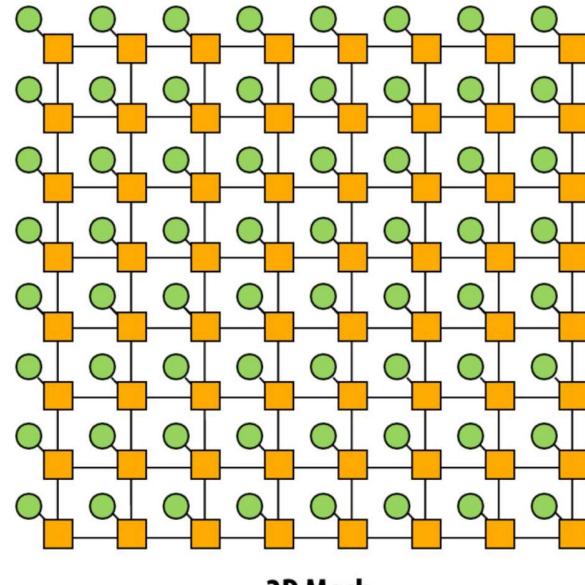


- **Four rings**
 - request
 - snoop
 - ack
 - data (32 bytes)
- **Six interconnect nodes: four “slices” of L3 cache + system agent + graphics**
- **Each bank of L3 connected to ring bus twice**
- **Theoretical peak BW from cores to L3 at 3.4 GHz is approx. 435 GB/sec**
 - When each core is accessing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

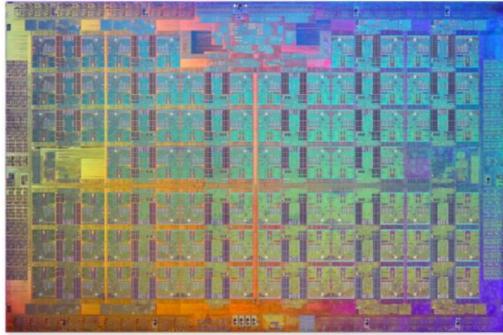
CMU 15-418/618, Spring 2017

Mesh

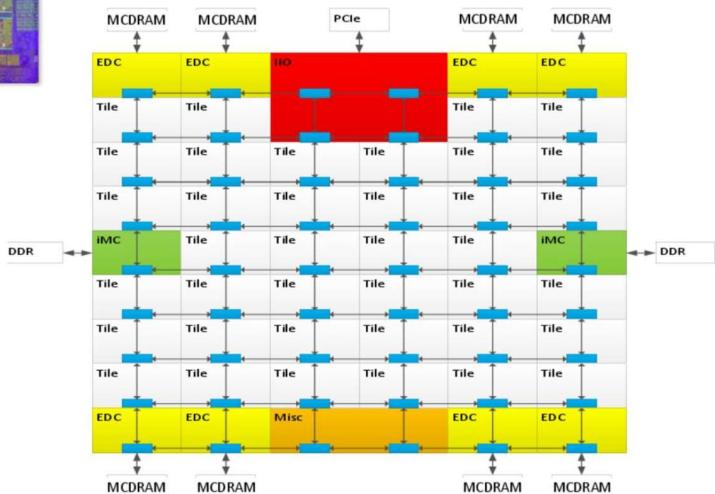
- Direct network
- Echoes locality in grid-based applications
- $O(N)$ cost
- Average latency: $O(\sqrt{N})$
- Easy to lay out on chip: fixed-length links
- Path diversity: many ways for message to travel from one node to another
- Used by:
 - Tilera processors
 - Prototype Intel chips



Xeon Phi (Knights Landing)



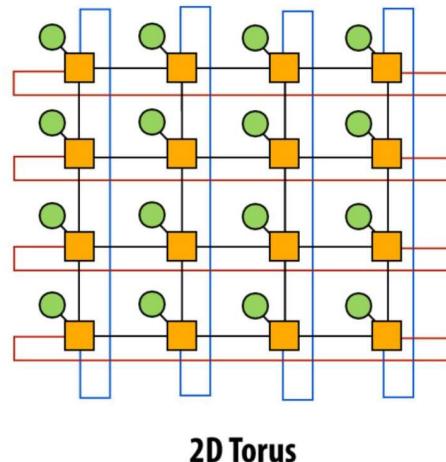
- 72 cores, arranged as 6 x 6 mesh of tiles (2 cores/tile)
- YX routing of messages:
 - Move in Y
 - "Turn"
 - Move in X



CMU 15-418/618, Spring 2017

Torus

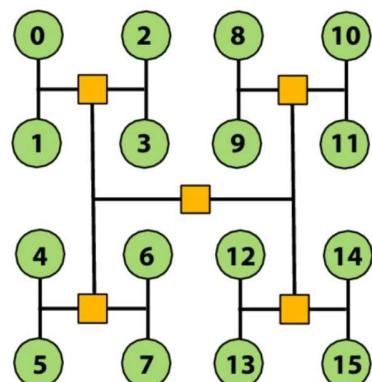
- Characteristics of mesh topology are different based on whether node is near edge or middle of network (torus topology introduces new links to avoid this problem)
- Still $O(N)$ cost, but higher cost than 2D grid
- Higher path diversity and bisection BW than mesh
- Higher complexity
 - Difficult to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



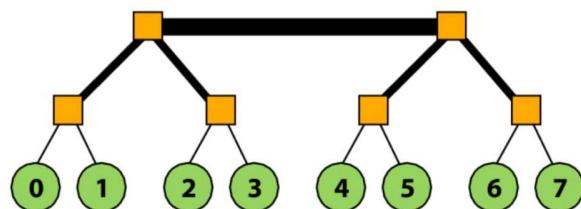
2D Torus

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)

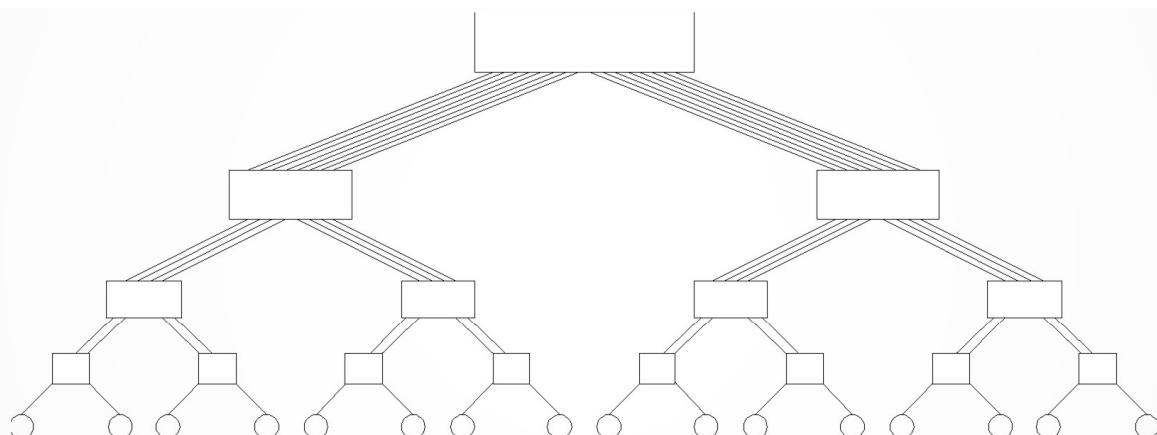


H-Tree



Fat Tree

CMU 15-418/618, Spring 2017

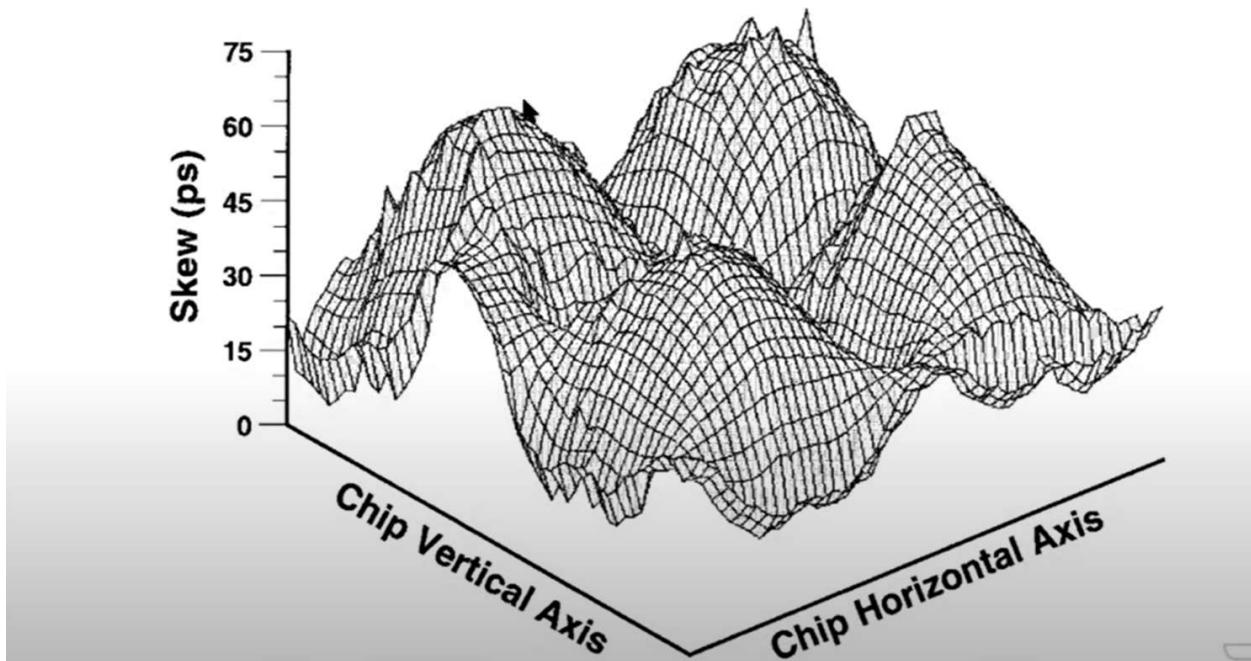


A fat tree network of 16 processing nodes.

Digression: H-trees are also used to distribute clock tick on the chip

Clock Skew Example

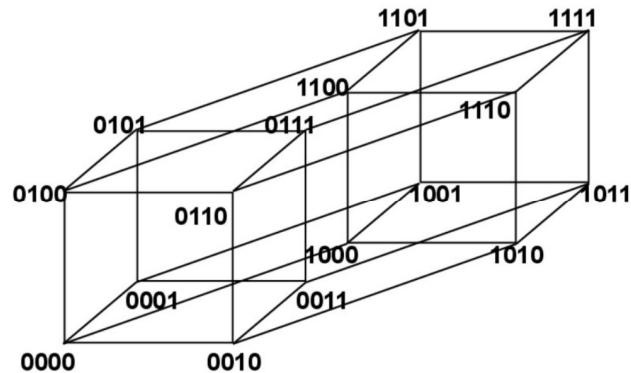
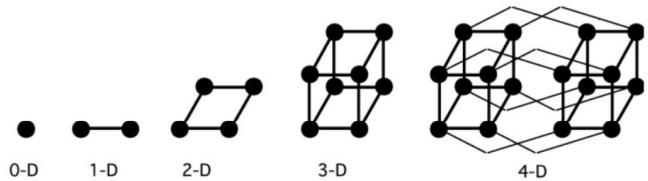
- Example of the **Alpha 21264** clock skew spatial distribution



[Screen grab from Prof. Onur Mutlu's lecture: <https://www.youtube.com/watch?v=4AcwBBwiNIQ>]

Hypercube

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



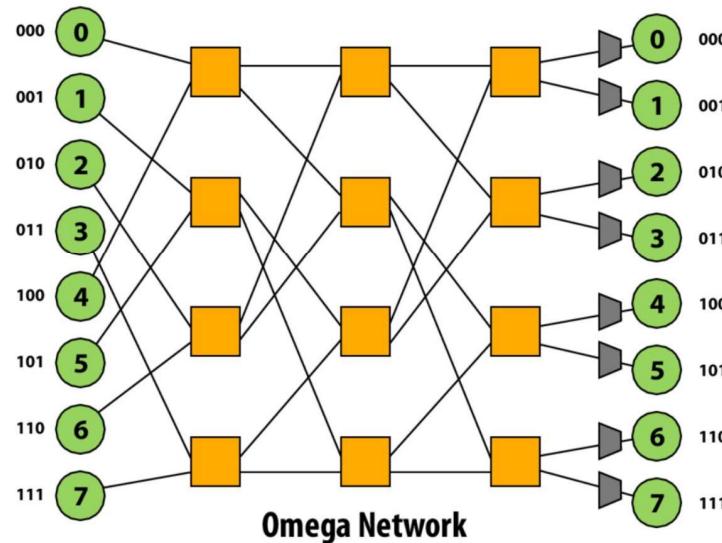
- 6D hypercube used in 64-core Cosmic Cube computer developed at Caltech in the 80s
- SGI Origin used a hypercube

CMU 15-418/618, Spring 2017

The labeling of the vertices of the hypercube is important here: they are used to devise the routing strategy thanks to the [E-Cube routing](#).

Multi-stage logarithmic

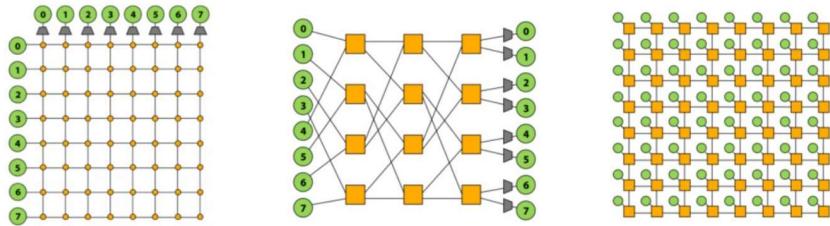
- Indirect network with multiple switches between terminals
- Cost: $O(N \lg N)$
- Latency: $O(\lg N)$
- Many variations: Omega, butterfly, Clos networks, etc...



CMU 15-418/618, Spring 2017

Clos is pronounced as “KLO”.

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	$O(1)$	$O(\lg N)$	$O(\sqrt{N})$ <small>(average)</small>

CMU 15-418/618, Spring 2017

Buffering and flow control:

Circuit switching vs. packet switching

- **Circuit switching sets up a full path (acquires all resources) between sender and receiver prior to sending a message**

- Establish route (reserve links) then send all data for message
- Higher bandwidth transmission (no per-packet link mgmt overhead)
- Does incur overhead to set up/tear down path
- Reserving links can result in low utilization



- **Packet switching makes routing decisions per packet**

- Route each packet individually (possibly over different network links)
- Opportunity to use link for a packet whenever link is idle
- Overhead due to dynamic switching logic during transmission
- No setup/tear down overhead



CMU 15-418/618, Spring 2017

Granularity of communication

■ Message

- Unit of transfer between network clients (e.g., cores, memory)
- Can be transmitted using many packets

■ Packet

- Unit of transfer for network
- Can be transmitted using multiple flits (will discuss later)

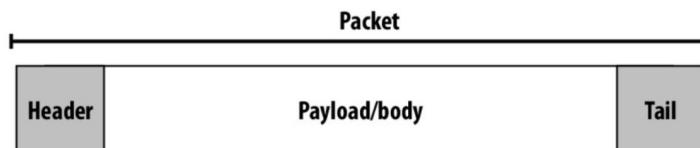
■ Flit (flow control digit)

- Packets broken into smaller units called “flits”
- Flit: (“flow control digit”) a unit of flow control in the network
- Flits become minimum granularity of routing/buffering

Packet format

- A packet consists of:

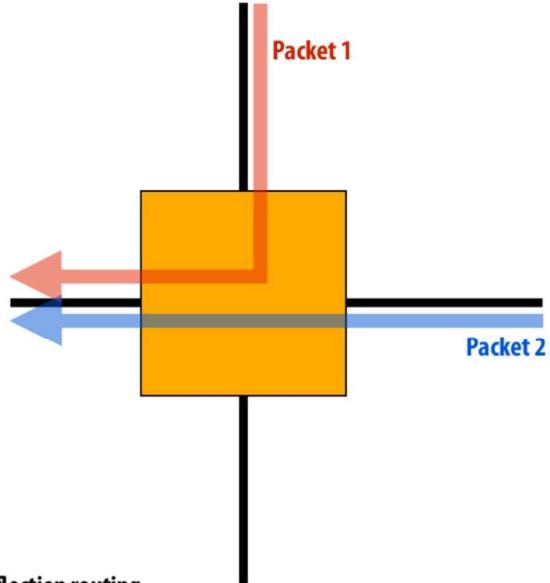
- Header:
 - Contains routing and control information
 - At start of packet so router can start forwarding early
- Payload/body: containing the data to be sent
- Tail
 - Contains control information, e.g., error code
 - Generally located at end of packet so it can be generated “on the way out”
(sender computes checksum, appends it to end of packet)



Handling contention

Scenario: two packets need to be routed onto the same outbound link at the same time

- Options:
 - Buffer one packet, send it over link later
 - Drop one packet
 - Reroute one packet (deflection)
- In this lecture: we only consider buffering *



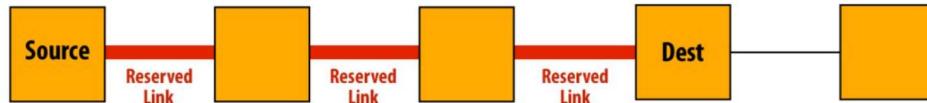
* But recent research has looked at using bufferless networks with deflection routing as a power-efficient interconnect for chip multiprocessors.

CMU 15-418/618, Spring 2017

Circuit-switched routing

■ High-granularity resource allocation

- Main idea: pre-allocate all resources (links across multiple switches) along entire network path for a message (“setup a flow”)



■ Costs

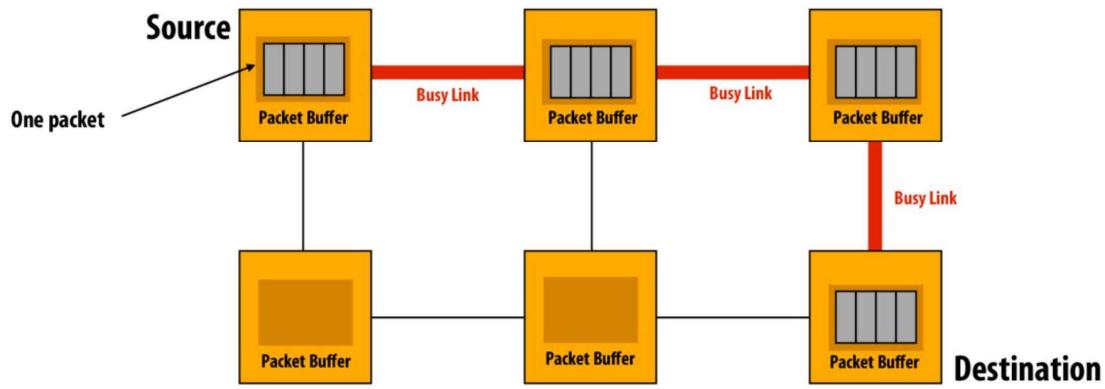
- Needs setup phase (“probe”) to set up the path (and to tear it down and release the resources when message complete)
- Lower link utilization. Transmission of two messages cannot share same link (even if some resources on a preallocated path are no longer utilized during a transmission)

■ Benefits

- No contention during transmission due to preallocation, so no need for buffering
- Arbitrary message sizes (once path is set up, send data until done)

Store-and-forward (packet-based routing)

- Packet copied entirely into network switch before moving to next node
- Flow control unit is an entire packet
 - Different packets from the same message can take different routes, but all data in a packet is transmitted over the same route
- Requires buffering for entire packet in each router
- High per-packet latency (latency = packet transmission time on link x network distance)

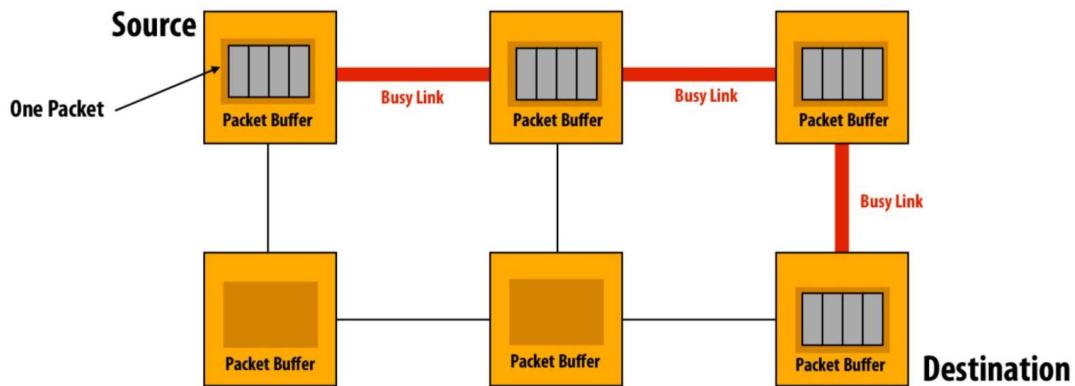


Note to students: in lecture this slide was animated and the final build shown here is not illustrative of store-and-forward routing concept (please refer to lecture video)

CMU 15-418/618, Spring 2017

Cut-through flow control (also packet-based)

- Switch starts forwarding data on next link as soon as packet header is received (header determines how much link bandwidth packet requires + where to route)
- Result: reduced transmission latency
 - Cut-through routing reduces to store-and-forward under high contention. Why?



Store and forward solution from previous slide: 3 hops x 4 units of time to transmit packet over a single link = 12 units of time

Cut-through solution: 3 steps of latency for head of packet to get to destination + 3 units of time for rest of packet = 6 units of time

Note to students: in lecture this slide was animated and the final build shown here is not illustrative of the cut-through routing concept (please refer to lecture video)

CMU 15-418/618, Spring 2017

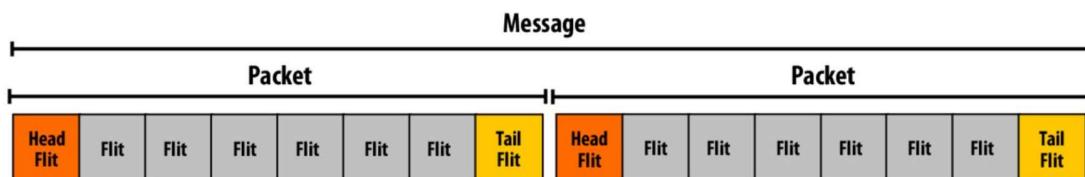
Cut-through flow control

- If output link is blocked (cannot transmit head), transmission of tail can continue
 - Worst case: entire message is absorbed into a buffer in a switch (cut-through flow control degenerates to store-and-forward in this case)
 - Requires switches to have buffering for entire packet, just like store-and-forward

Wormhole flow control

■ Flit (flow control digit)

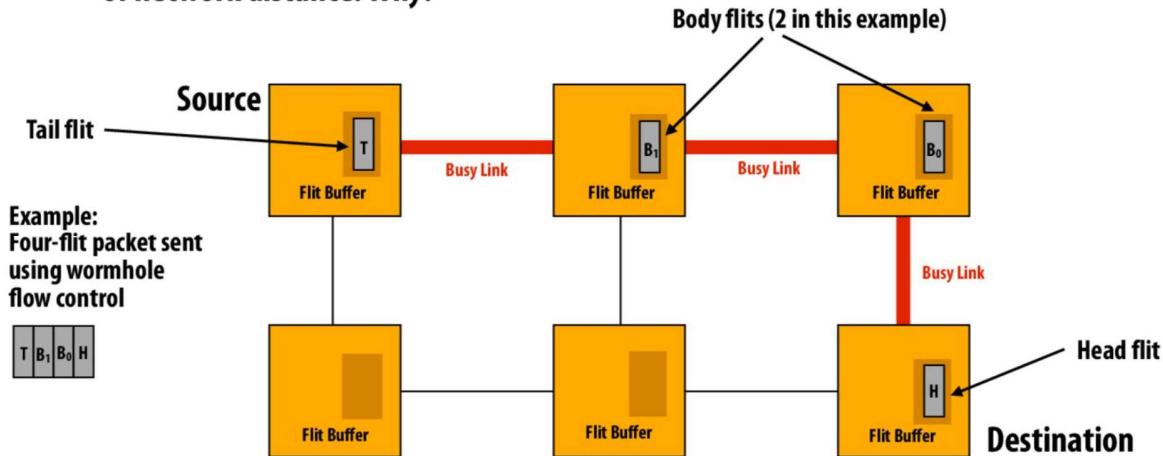
- Packets broken into smaller units called “flits”
- Flit: (“flow control digit”) a unit of flow control in the network
- Flits become minimum granularity of routing/buffering
 - Recall: up until now, packets were the granularity of transfer AND flow control and buffering (store-and-forward, cut-through routing)



Wormhole flow control

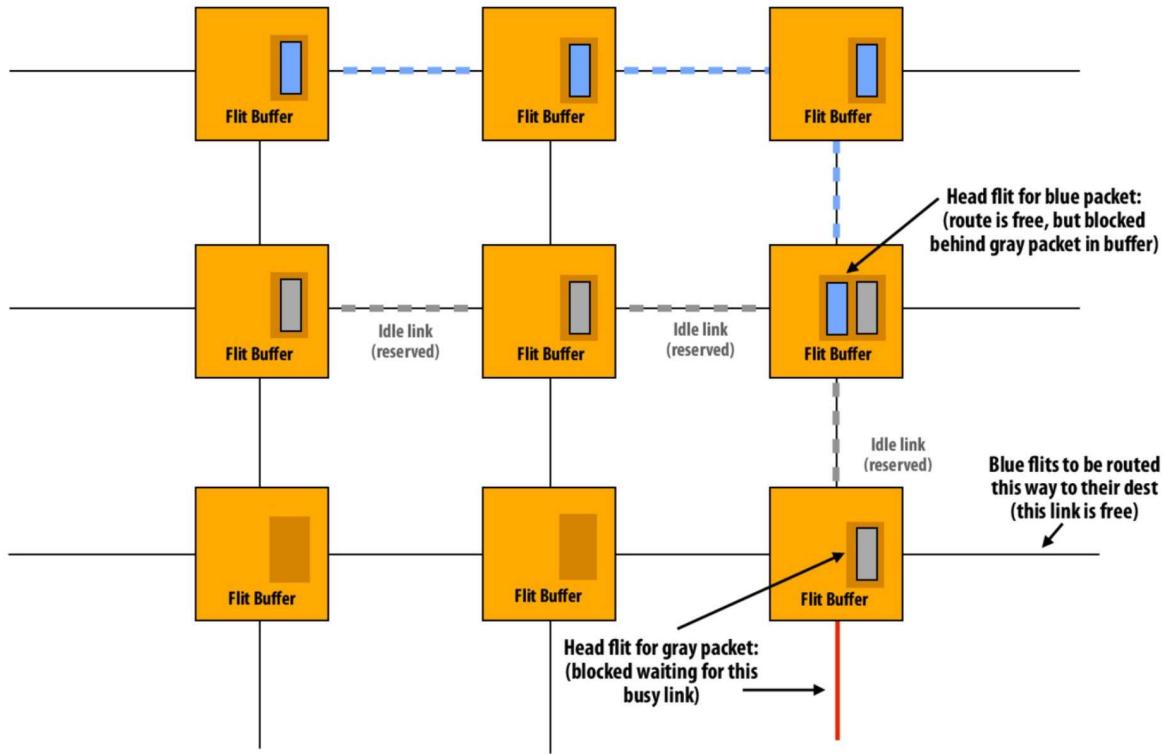


- Routing information only in head flit
- Body flits follows head, tail flit flows body
- If head flit blocks, rest of packet stops
- Completely pipelined transmission
 - For long messages, latency is almost entirely independent of network distance. Why?



CMU 15-418/618, Spring 2017

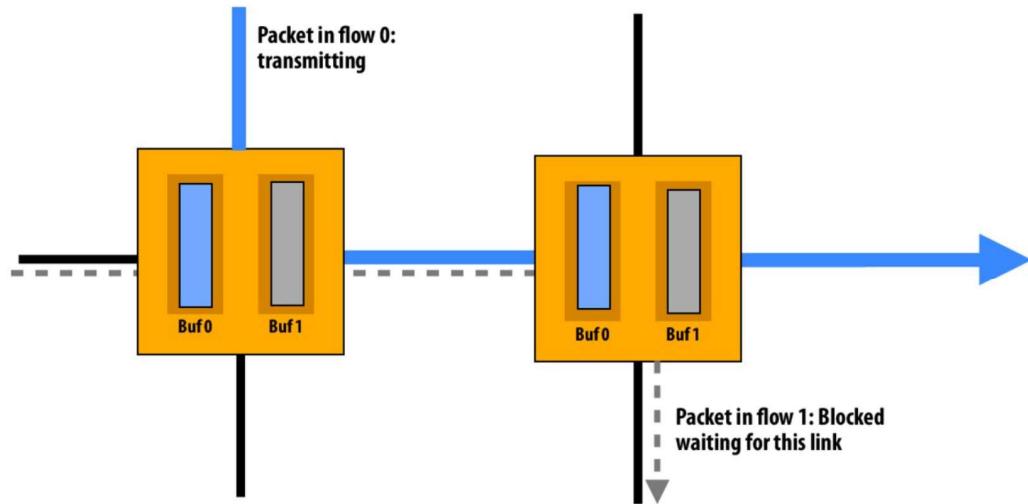
Problem: head-of-line blocking



CMU 15-418/618, Spring 2017

Virtual channel flow control

- Multiplex multiple operations over single physical channel
- Divide switch's input buffer into multiple buffers sharing a single physical channel
- Reduces head-of-line blocking



See "Virtual Channel Flow Control," [Dally ISCA 1990]

CMU 15-418/618, Spring 2017

Other uses of virtual channels

■ Deadlock avoidance

- Can be used to break cyclic dependency of resources
- Prevent cycles by ensuring requests and responses use different virtual channels
- “Escape” VCs: retain at least one virtual channel that uses deadlock-free routing

■ Prioritization of traffic classes

- Provide quality-of-service guarantees
- Some virtual channels have higher priority than others

Current research topics

- **Energy efficiency of interconnections**
 - Interconnect can be energy intensive (~35% of total chip power in MIT RAW research processor)
 - Bufferless networks
 - Other techniques: turn on/off regions of network, use fast and slow networks
- **Prioritization and quality-of-service guarantees**
 - Prioritize packets to improve multi-processor performance (e.g., some applications may be more sensitive to network performance than others)
 - Throttle endpoints (e.g., cores) based on network feedback
- **New/emerging technologies**
 - Die stacking (3D chips)
 - Photonic networks-on-chip (use optical waveguides instead of wires)
 - Reconfigurable devices (FPGAs): create custom interconnects tailored to application (see CMU projects: CONNECT, CoRAM, Shrinkwrap)

Summary

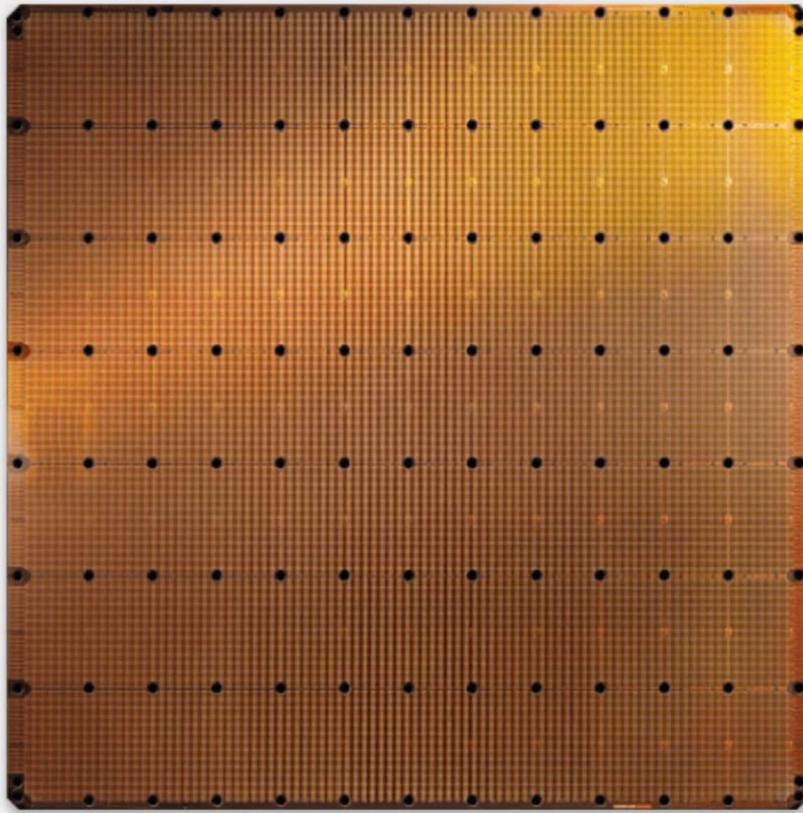
- The performance of the interconnection network in a modern multi-processor is critical to overall system performance
 - Buses do not scale to many nodes
 - Historically interconnect was off-chip network connecting sockets, boards, racks
 - Today, all these issues apply to the design of on-chip networks
- Network topologies differ in performance, cost, complexity tradeoffs
 - e.g., crossbar, ring, mesh, torus, multi-stage network, fat tree, hypercube
- Challenge: efficiently routing data through network
 - Interconnect is a precious resource (communication is expensive!)
 - Flit-based flow control: fine-grained flow control to make good use of available link bandwidth
 - If interested, much more to learn about (not discussed in this class): ensuring quality-of-service, prioritization, reliability, deadlock, livelock, etc.

CMU 15-418/618, Spring 2017

Case study: Cerebras Processor (wafer scale):

Cluster-Scale Performance on a
Single Chip

(<https://www.cerebras.net/product-chip/>)



Cerebras WSE-2

46,225mm² Silicon
2.6 Trillion transistors

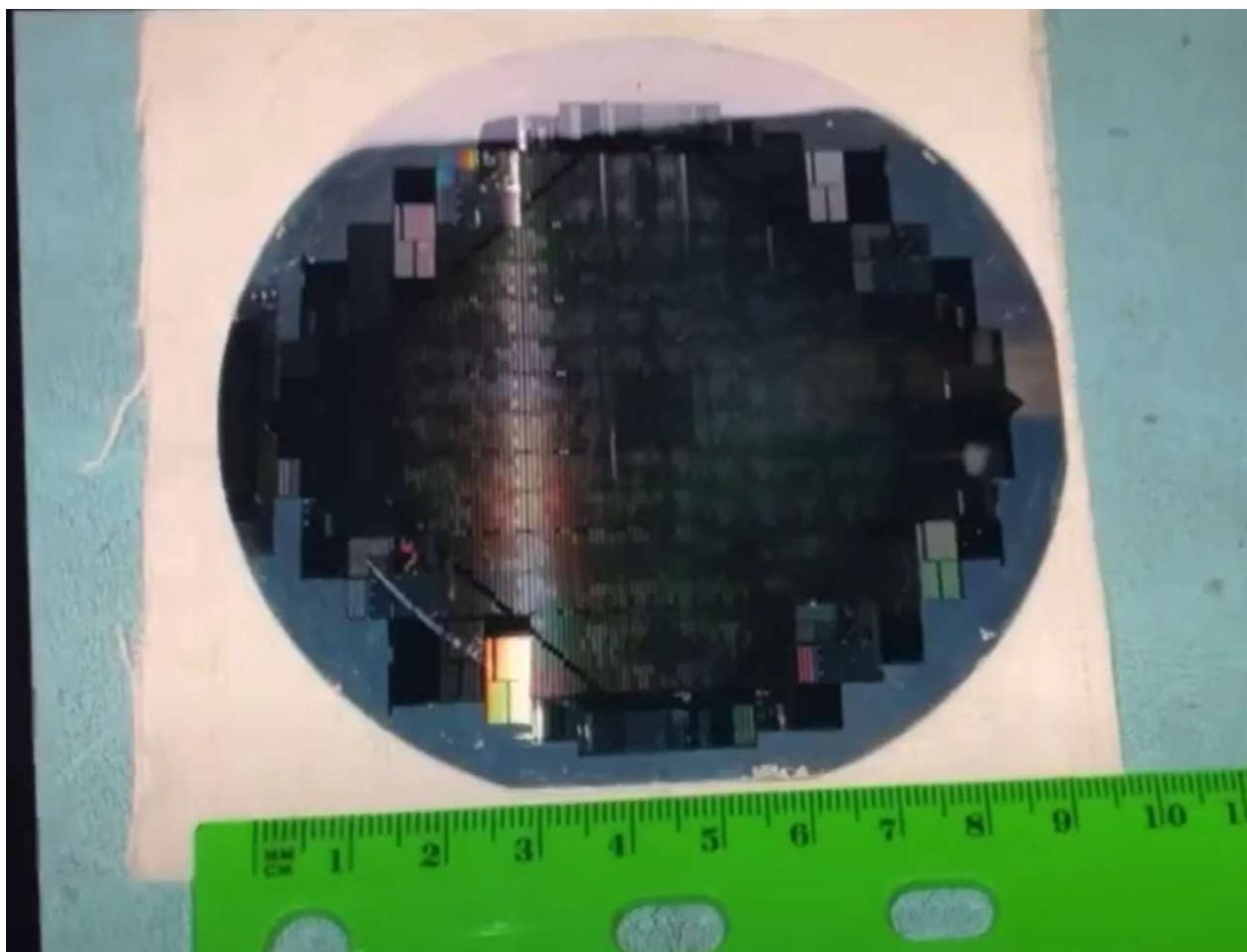
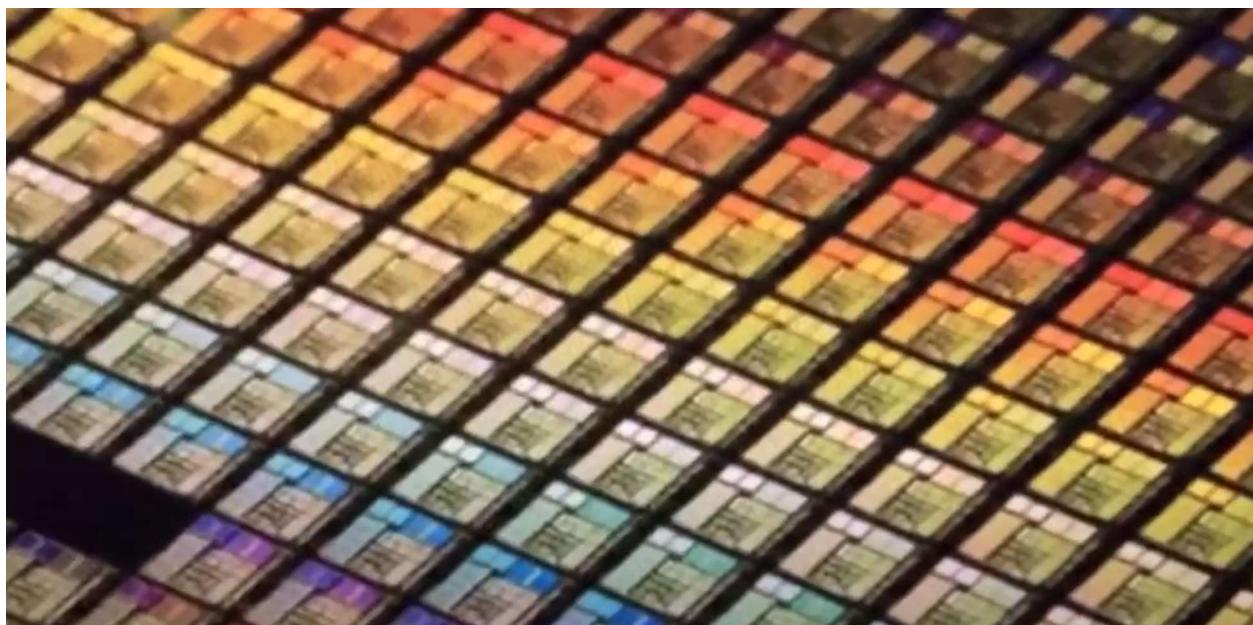
Largest GPU

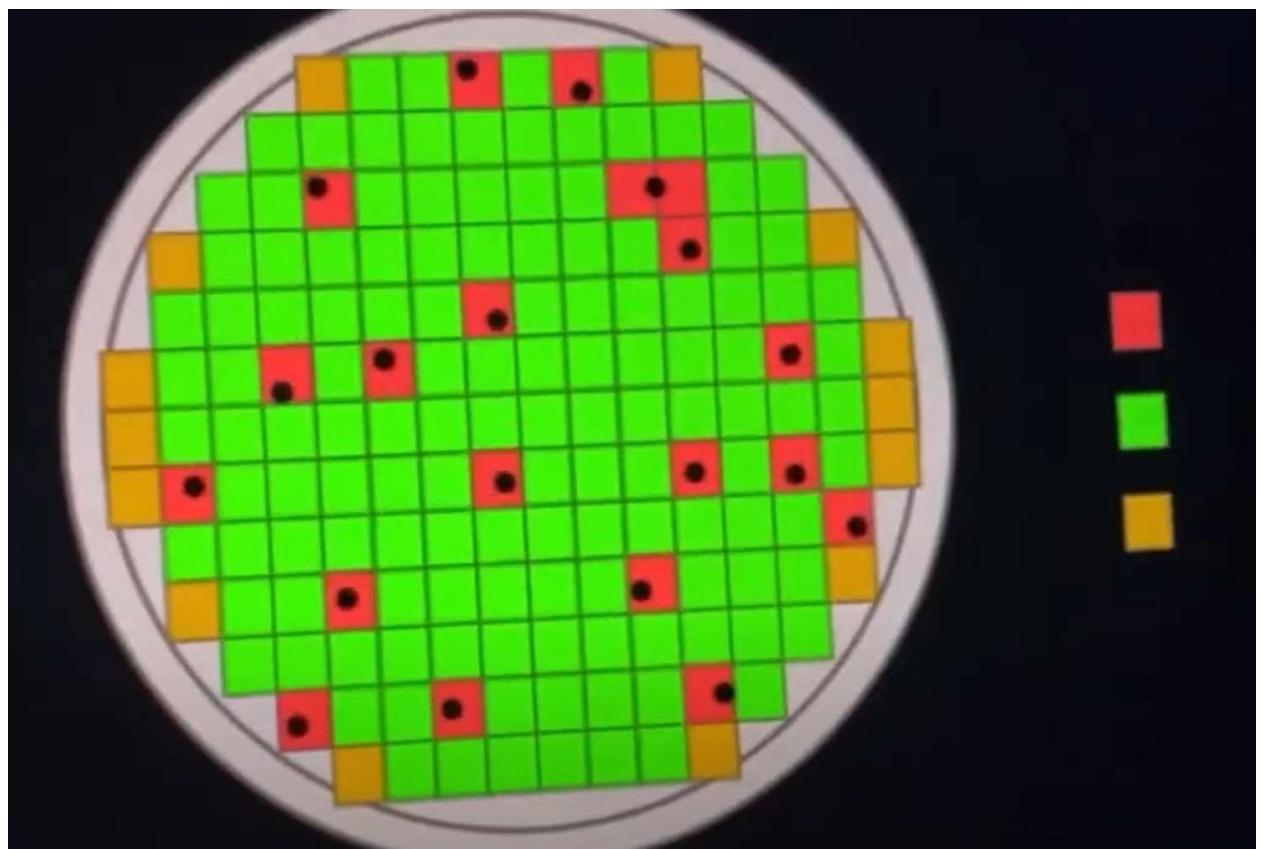
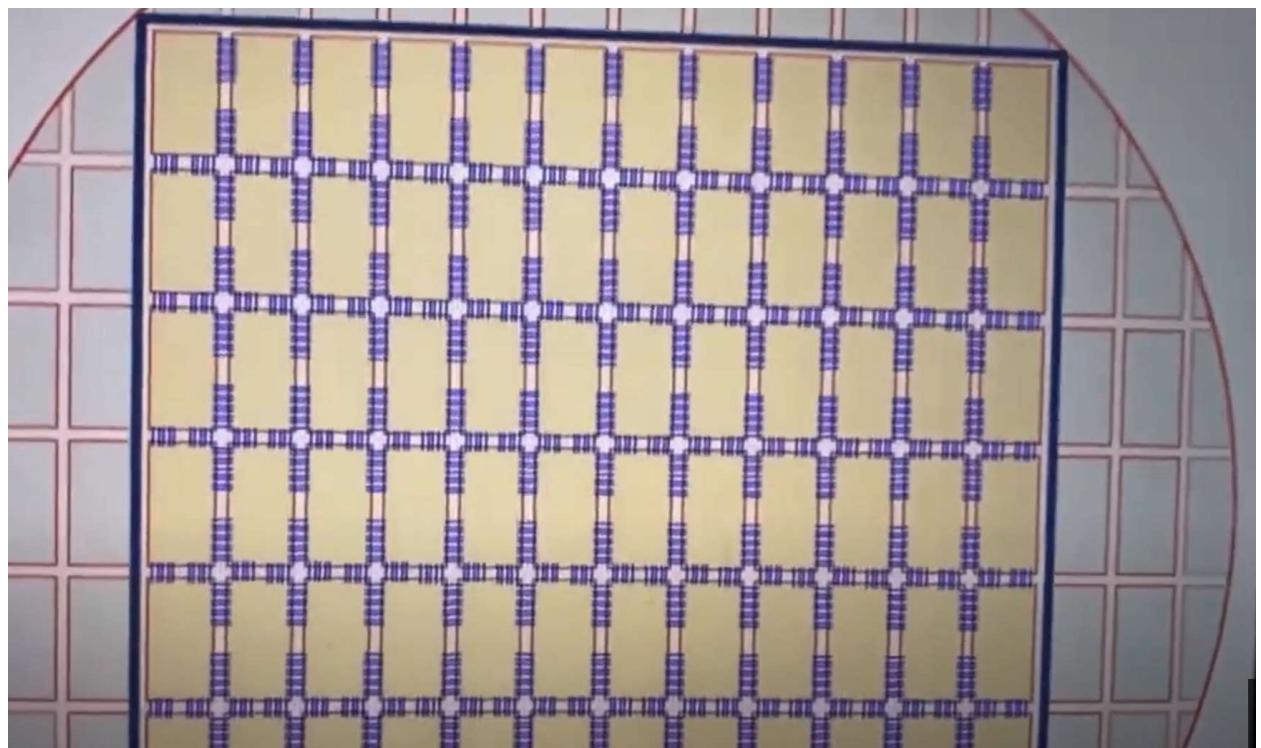
826mm² Silicon
54.2 Billion transistors

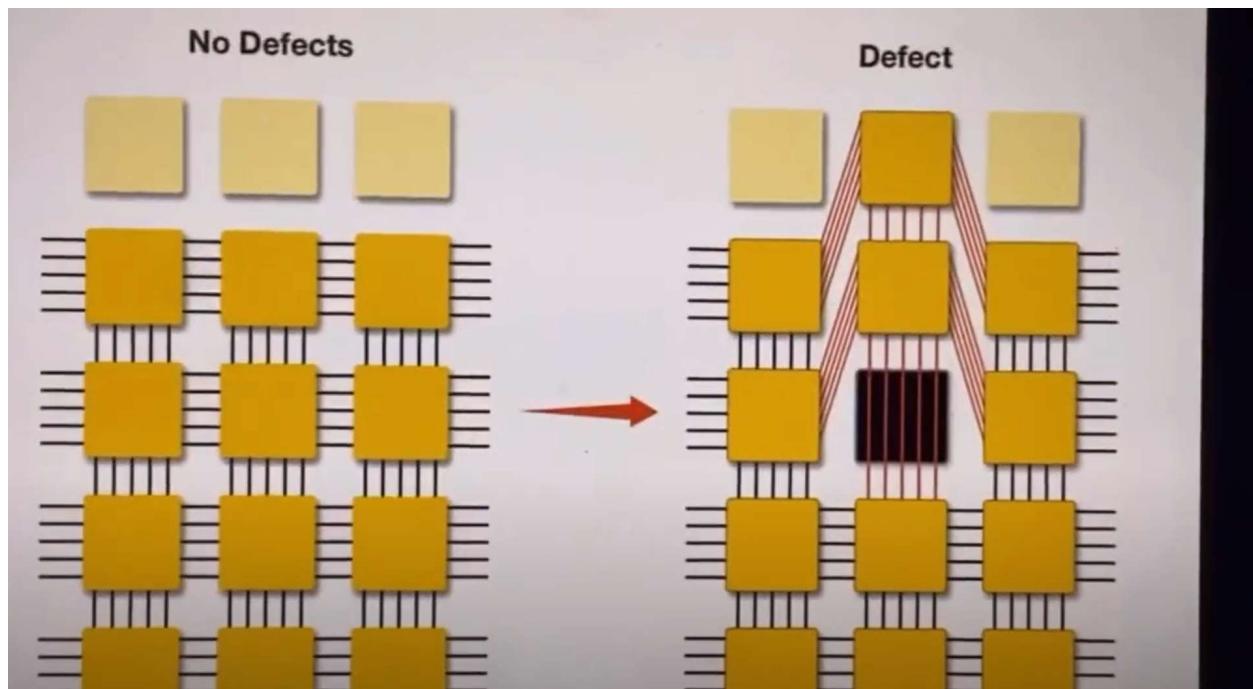
The Wafer-Scale Advantage

The Wafer-Scale Engine delivers unparalleled performance with no trade-offs

	WSE-2	A100	Cerebras Advantage
Chip Size	46,225 mm ²	826 mm ²	56 X
Cores	850,000	6912 + 432	123X
On-chip memory	40 Gigabytes	40 Megabytes	1,000 X
Memory bandwidth	20 Petabytes/sec	1.6 Terabytes/sec	12,733 X
Fabric bandwidth	220 Petabits/sec	4.8 Terabits/sec	45,833 X







[Above 5 pictures screen grabbed from: <https://www.youtube.com/watch?v=LjJaHflemKU>]

You can see more detailed discussion of this chip at: https://www.youtube.com/watch?v=8i1_Ru5siXc