

2023年度(令和5年)版


Ver. 2023-11-24a

Course number: CSC.T363



コンピュータアーキテクチャ Computer Architecture

12. ベクタ、SIMDにおけるデータレベル並列性 Data-Level Parallelism in Vector and SIMD



www.arch.cs.titech.ac.jp/lecture/CA/
Tue 13:30-15:10, 15:25-17:05
Fri 13:30-15:10



吉瀬 謙二 情報工学系
Kenji Kise, Department of Computer Science
kise_at_c.titech.ac.jp

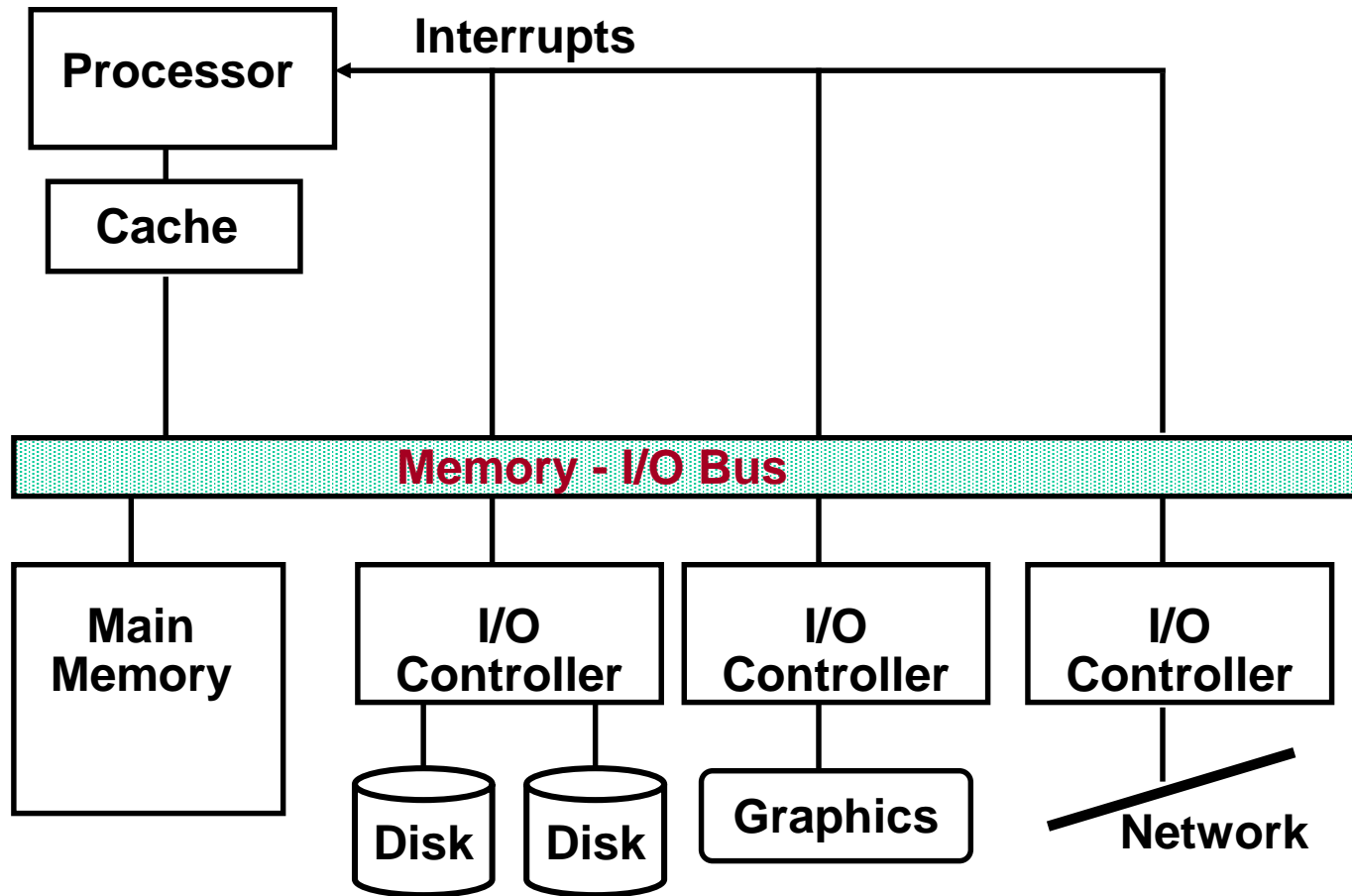
Final Exam



- <https://www.arch.cs.titech.ac.jp/lecture/CA/>
- **2023-12-01 (13:30-15:10)**
final exam in the computer room of the department of
computer science



A Typical I/O System and interrupts



Communication of I/O Devices and Processor (1)

- How the processor directs the I/O devices
 - **Memory-mapped I/O**
 - Portions of the high-order memory address space are assigned to each I/O device
 - Read and writes to those memory addresses are interpreted as commands to the I/O devices
 - Load/stores to the I/O address space can only be done by the OS
 - **Special I/O instructions**

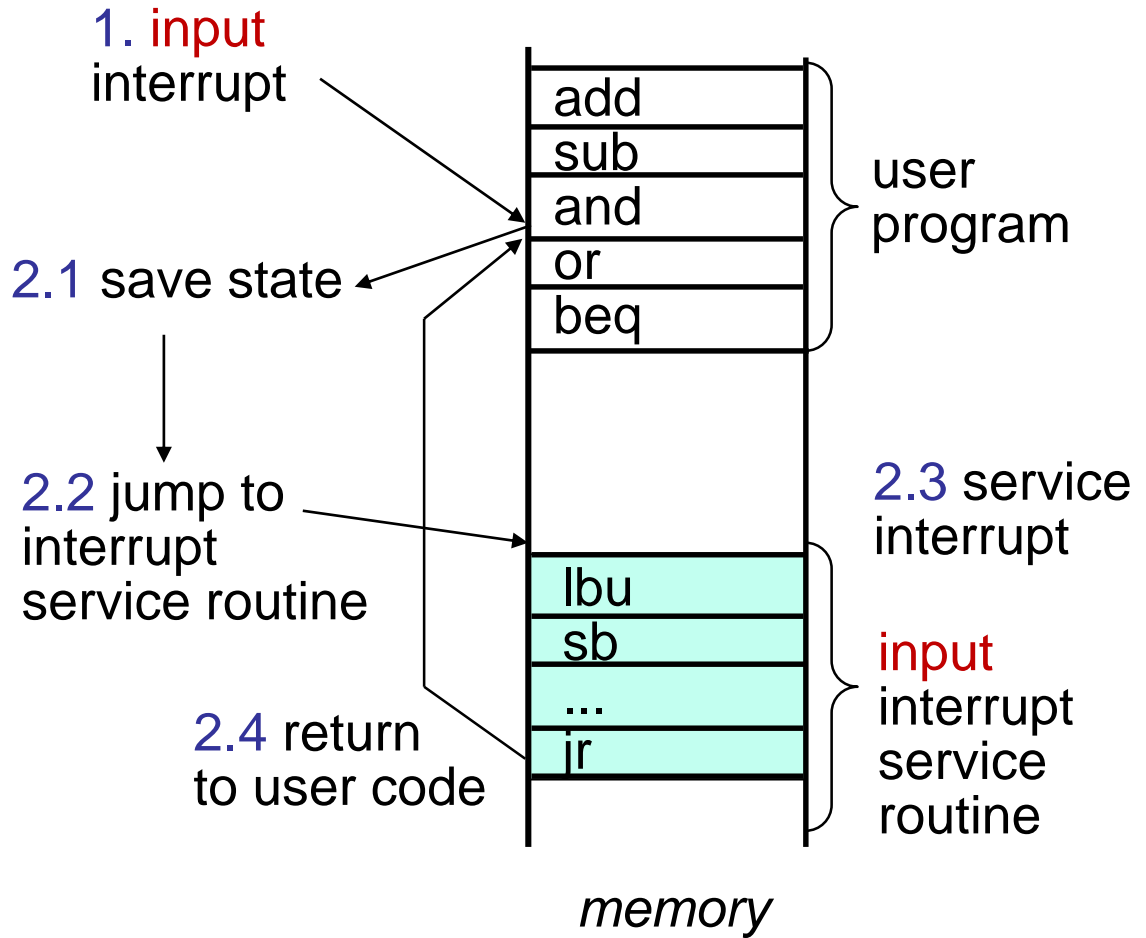
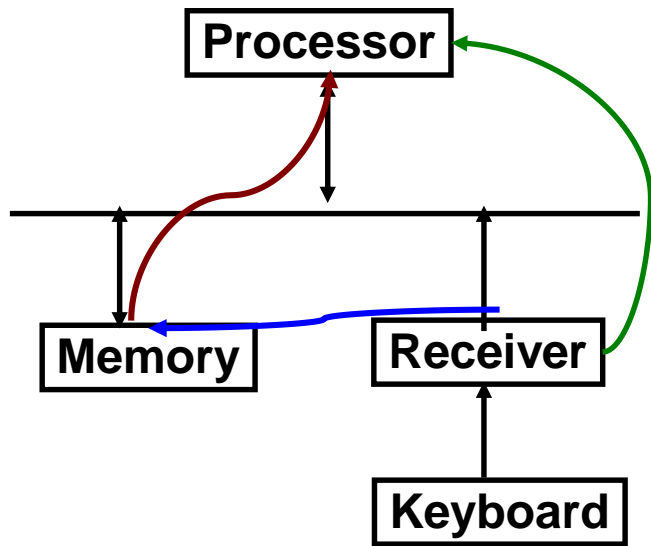


Communication of I/O Devices and Processor (2)

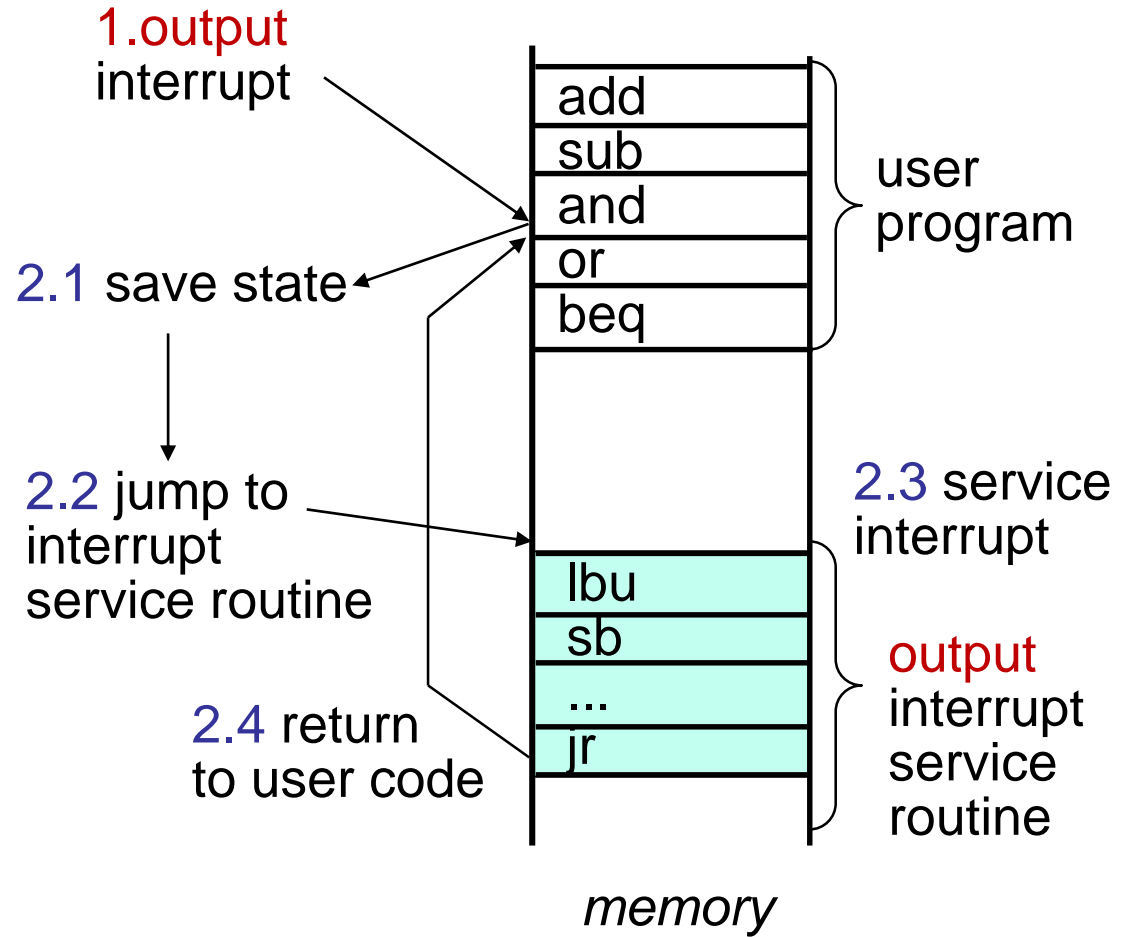
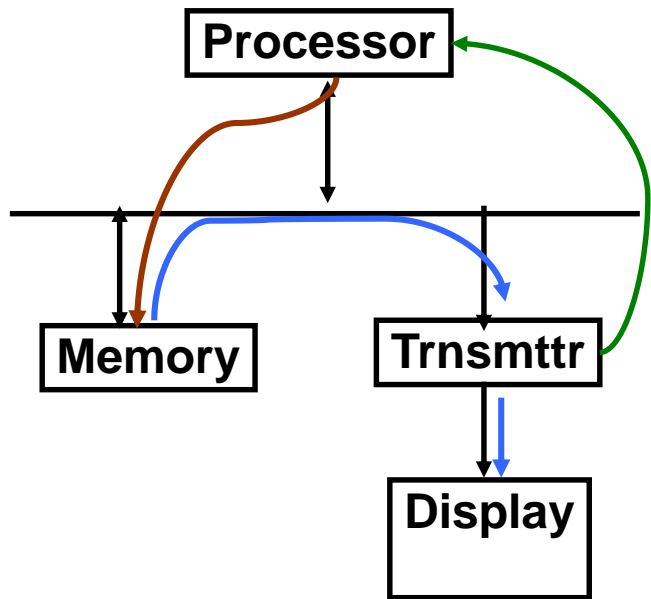
- How the I/O device communicates with the processor
 - **Polling** – the processor periodically checks the status of an I/O device to determine its need for service
 - Processor is totally in control – but does all the work
 - Can waste a lot of processor time due to speed differences
 - **Interrupt-driven I/O** – the I/O device issues an interrupts to the processor to indicate that it needs attention



Interrupt-Driven Input



Interrupt-Driven Output



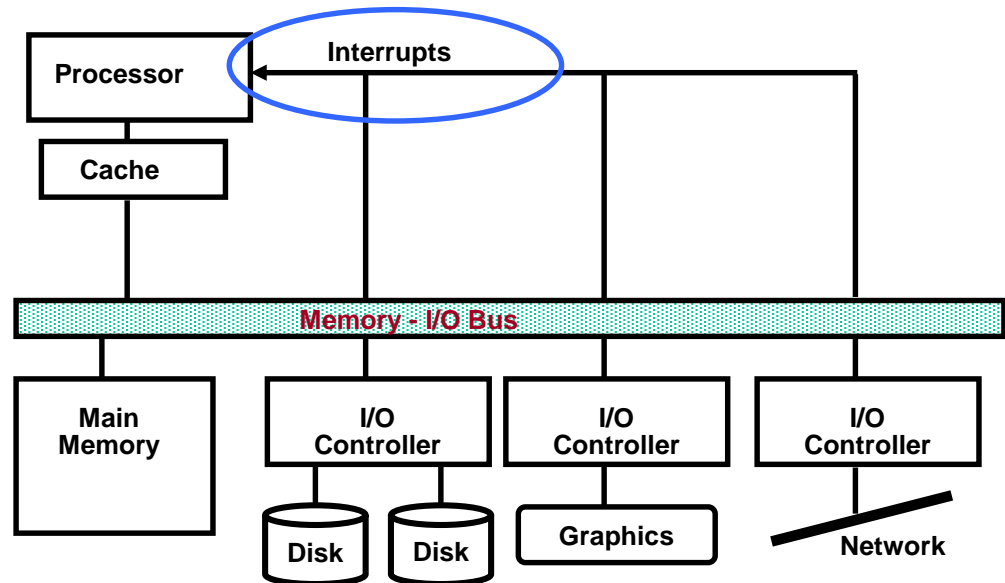
Interrupt-Driven I/O

- An I/O interrupt is **asynchronous**
 - Is not associated with any instruction so doesn't prevent any instruction from completing
 - You can pick your own convenient point to handle the interrupt
- With I/O interrupts
 - Need a way to identify the device generating the interrupt
 - Can have different urgencies (so may need to be prioritized)
- **Advantages** of using interrupts
 - No need to continuously poll for an I/O event; user program progress is only suspended during the actual transfer of I/O data to/from user memory space
- **Disadvantage** – special hardware is needed to
 - Cause an interrupt (I/O device) and detect an interrupt and save the necessary information to resume normal processing after servicing the interrupt (processor)



Direct Memory Access (DMA)

- For high-bandwidth devices (like disks) **interrupt-driven I/O** would consume a lot of processor cycles
- DMA – the I/O controller has the ability to transfer data **directly** to/from the memory without involving the processor
- There may be multiple DMA devices in one system



Direct Memory Access (DMA) how to?

1. The processor initiates the DMA transfer by supplying
 1. the I/O device address
 2. the operation to be performed
 3. the memory address destination/source
 4. the number of bytes to transfer.
2. The I/O DMA controller manages the entire transfer arbitrating for the bus
3. When the DMA transfer is complete, the I/O controller **interrupts** the processor to let it know that the transfer is complete

- **Cache Coherence**



I/O and the Operating System

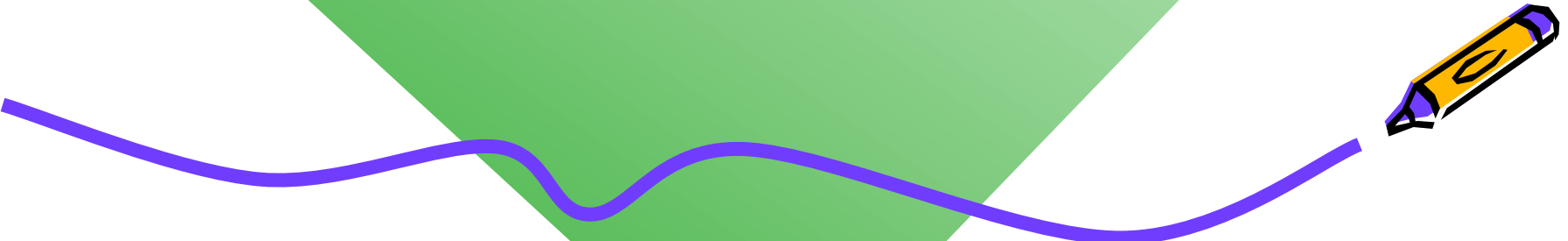
- The operating system acts as the interface between the I/O hardware and the program requesting I/O
 - To protect the **shared I/O resources**, the user program is not allowed to communicate directly with the I/O device
- Thus OS must be able to give commands to I/O devices, handle interrupts generated by I/O devices, provide fair access to the shared I/O resources, and schedule I/O requests to enhance system throughput
 - I/O interrupts result in a transfer of processor control to the **supervisor (OS) process**





コンピュータアーキテクチャ Computer Architecture

13. 相互接続ネットワーク、マルチプロセッサ、マルチコア Interconnection Network, Multiprocessors and Multicore



www.arch.cs.titech.ac.jp/lecture/CA/
Tue 13:30-15:10, 15:25-17:05
Fri 13:30-15:10

吉瀬 謙二 情報工学系
Kenji Kise, Department of Computer Science
kise_at_c.titech.ac.jp

From multi-core era to many-core era

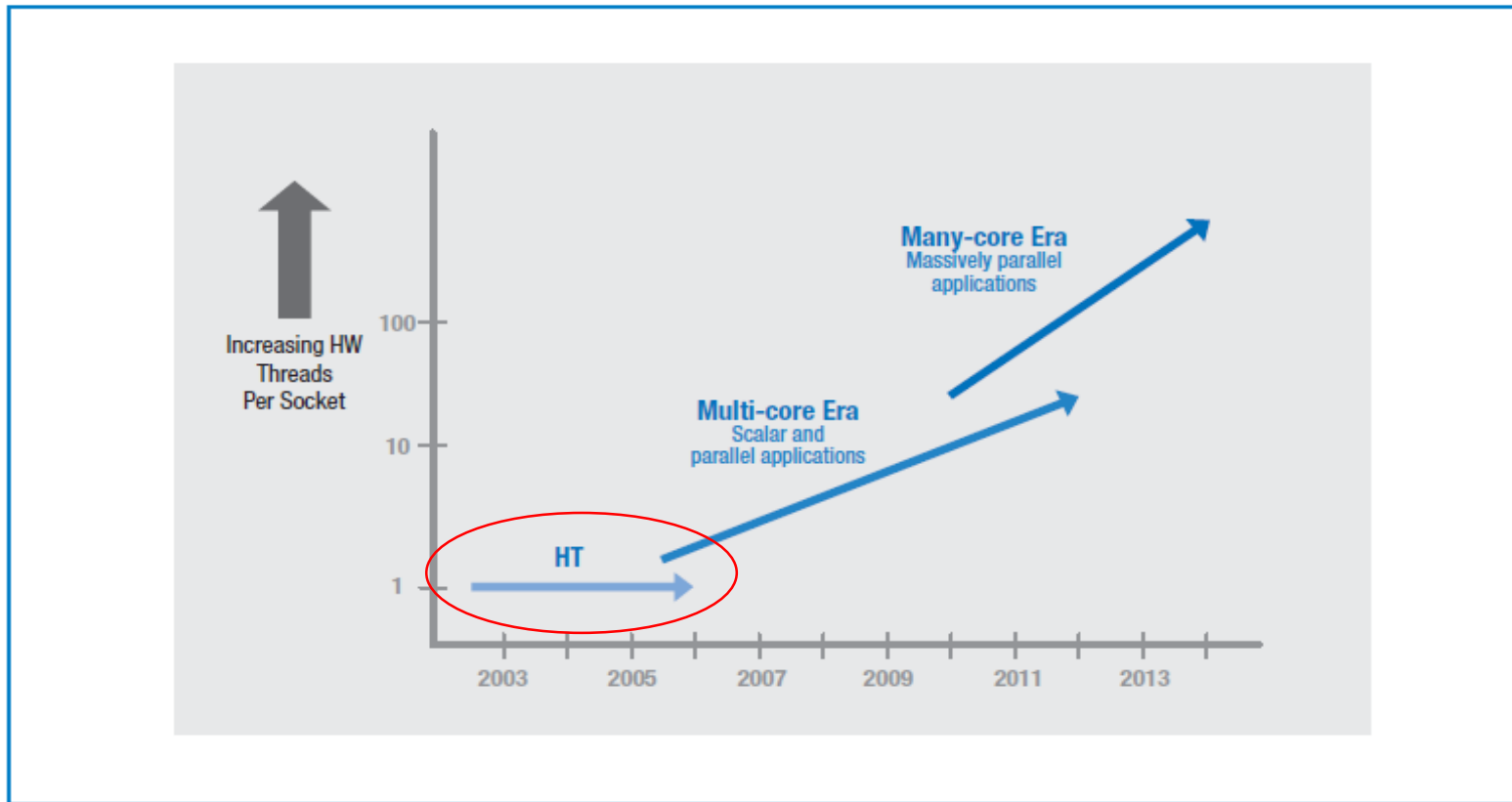
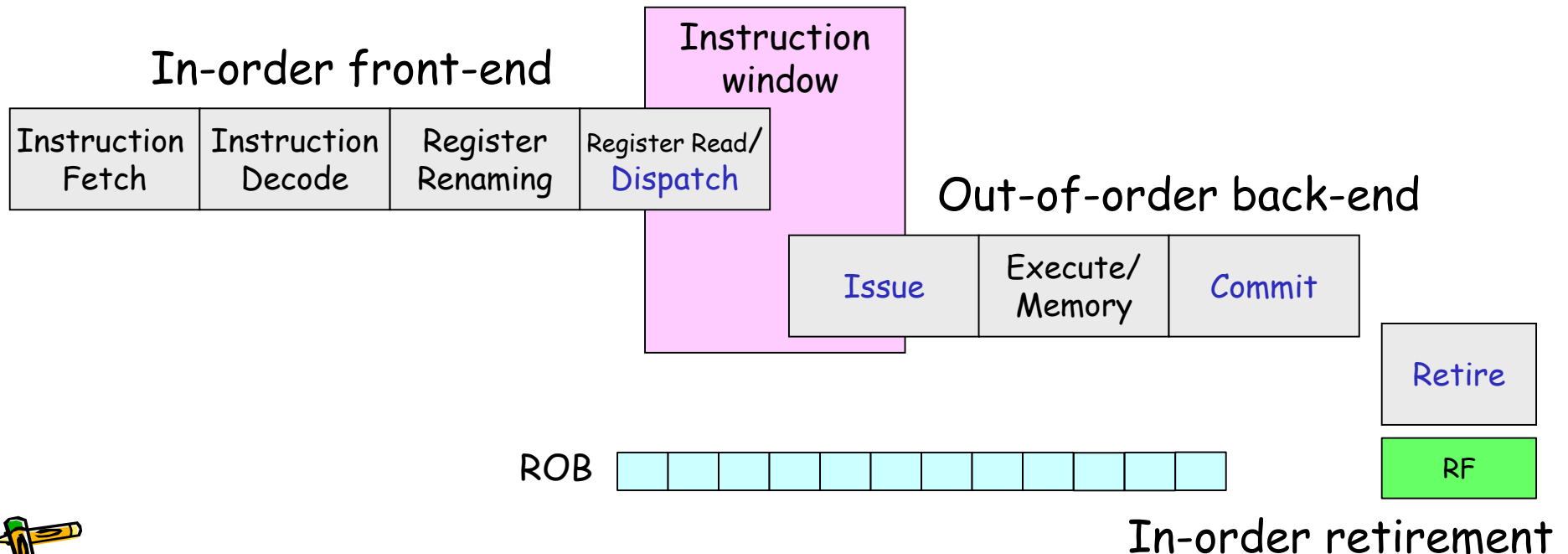


Figure 1: Current and expected eras of Intel® processor architectures

Platform 2015: Intel® Processor and Platform Evolution for the Next Decade, 2005

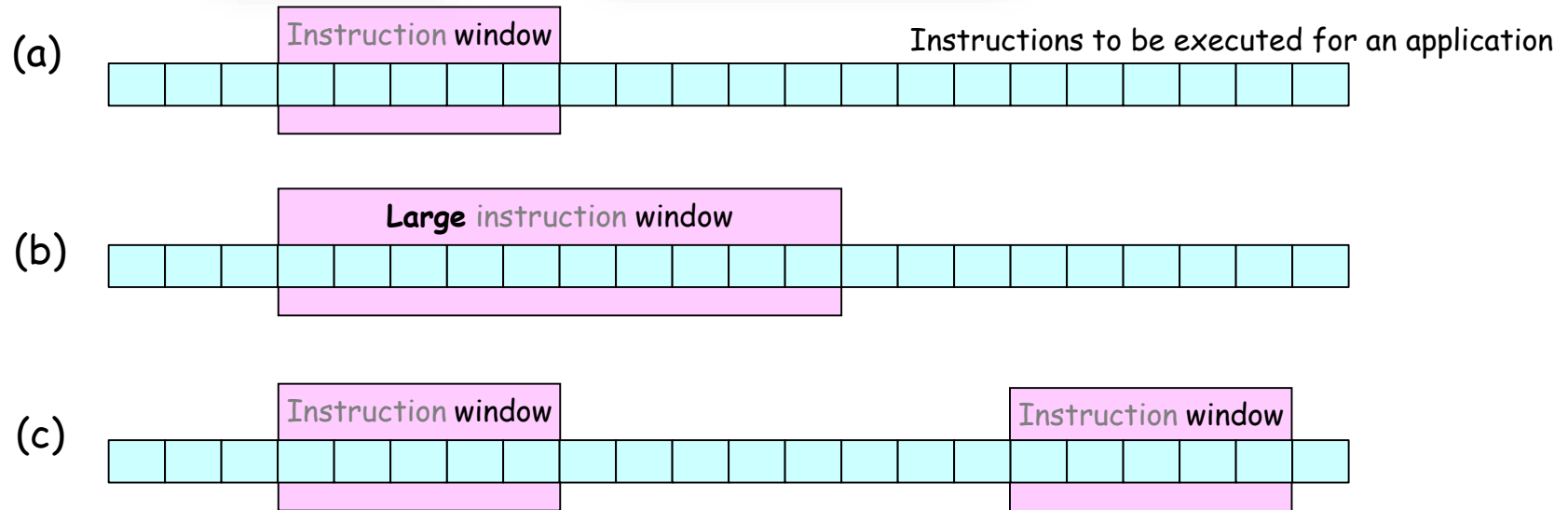
Instruction pipeline of **out-of-order (OoO) execution**

- Allocating instructions to instruction window is called dispatch
- Issue or fire wakes up instructions and their executions begin
- In commit stage, the computed values are written back to ROB (reorder buffer)
- The last stage is called retire or graduate. The result is written back to register file (architectural register file) using a logical register number.



Aside: what is a window?

- A window is a space in the wall of a building or in the side of a vehicle, which has glass in it so that light can come in and you can see out. (Collins)



Pollack's Rule

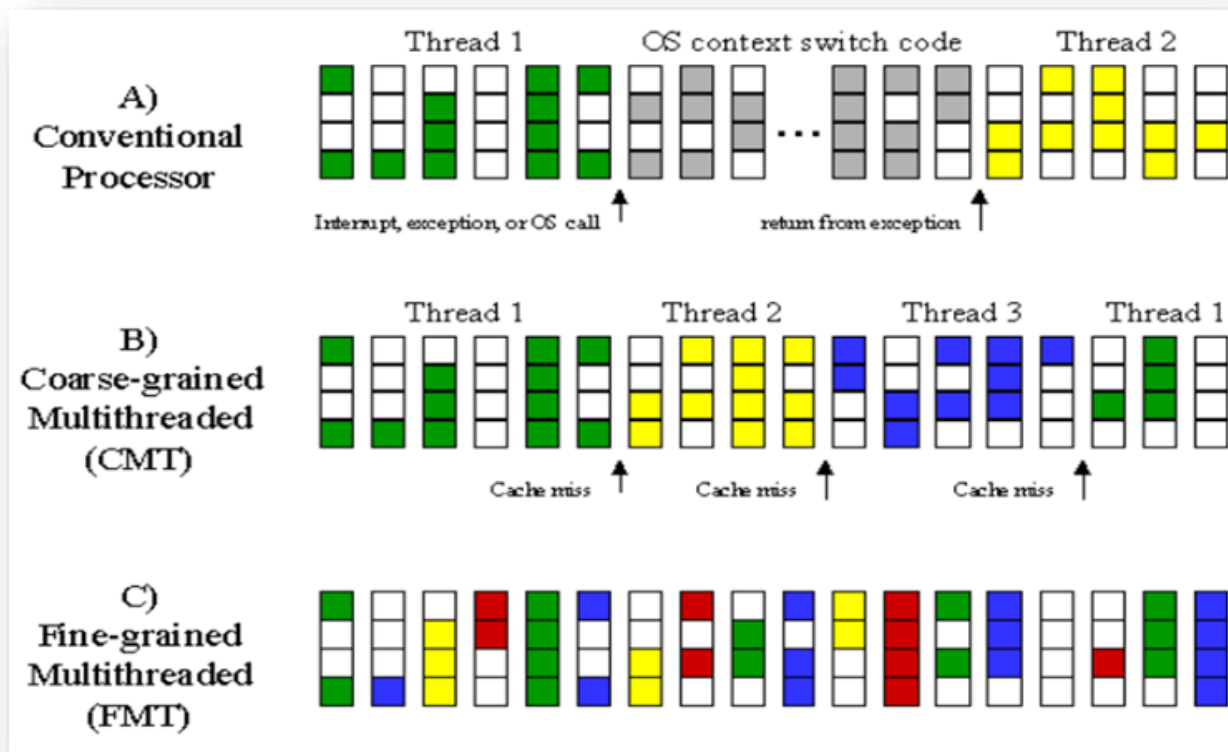


- Pollack's Rule states that microprocessor "performance increase due to microarchitecture advances is roughly proportional to the square root of the increase in complexity". Complexity in this context means processor logic, i.e. its area.
- Superscalar, vector
 - Instruction level parallelism, data level parallelism



Multithreading (1/2)

- During a branch miss recovery and access to the main memory by a cache miss, ALUs have no jobs to do and have to be idle.
- Executing **multiple independent threads (programs)** will mitigate the overhead.
- They are called **coarse-** and **fine-grained** multithreaded processors having multiple architecture states.

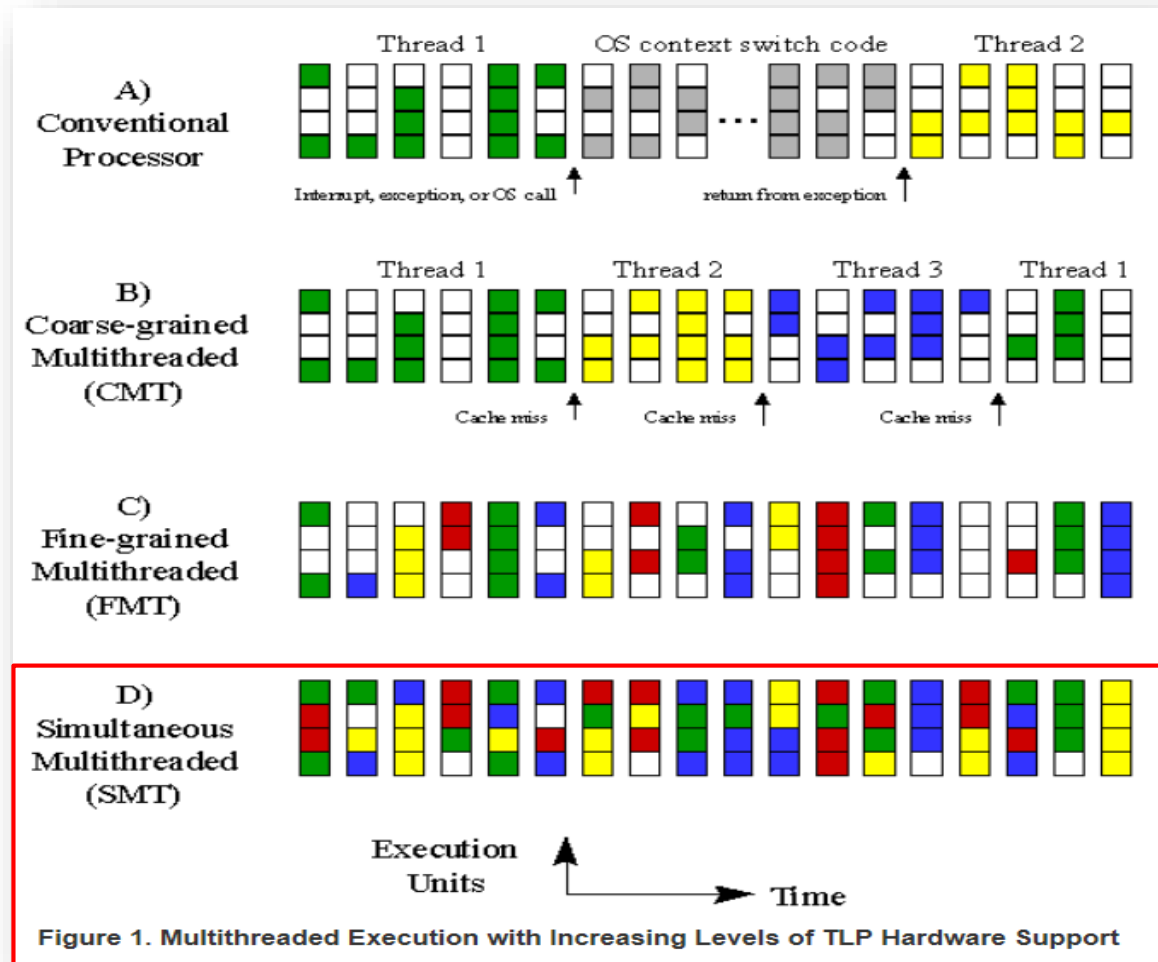


Execution Units ↑
Time →



Multithreading (2/2)

- Simultaneous Multithreading (SMT) can improve hardware resource usage.



From multi-core era to many-core era

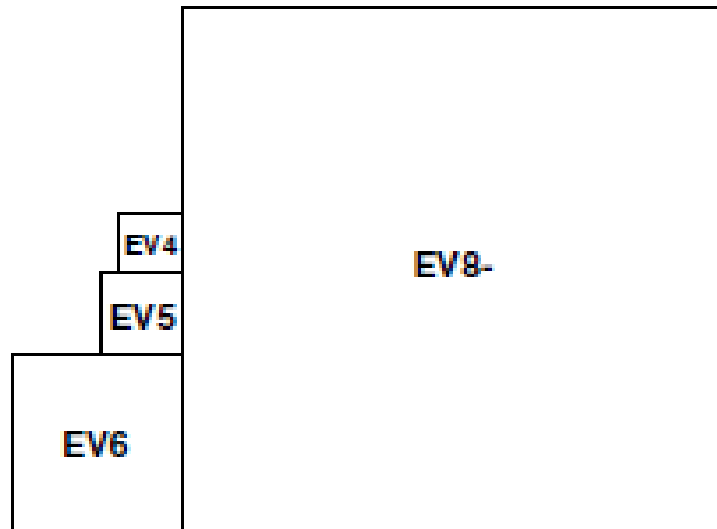


Figure 1. Relative sizes of the cores used in the study

Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction, MICRO-36

From multi-core era to many-core era

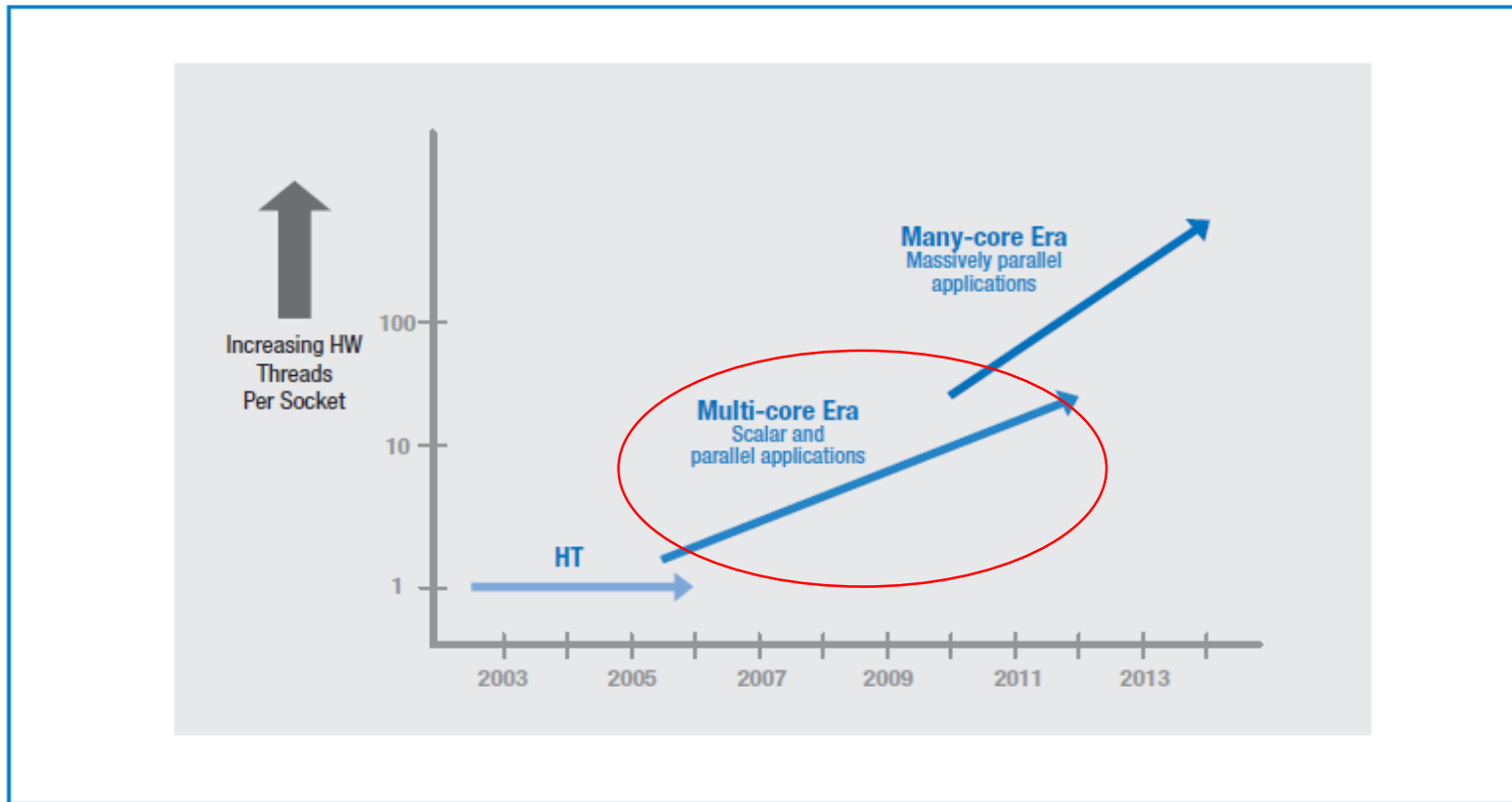
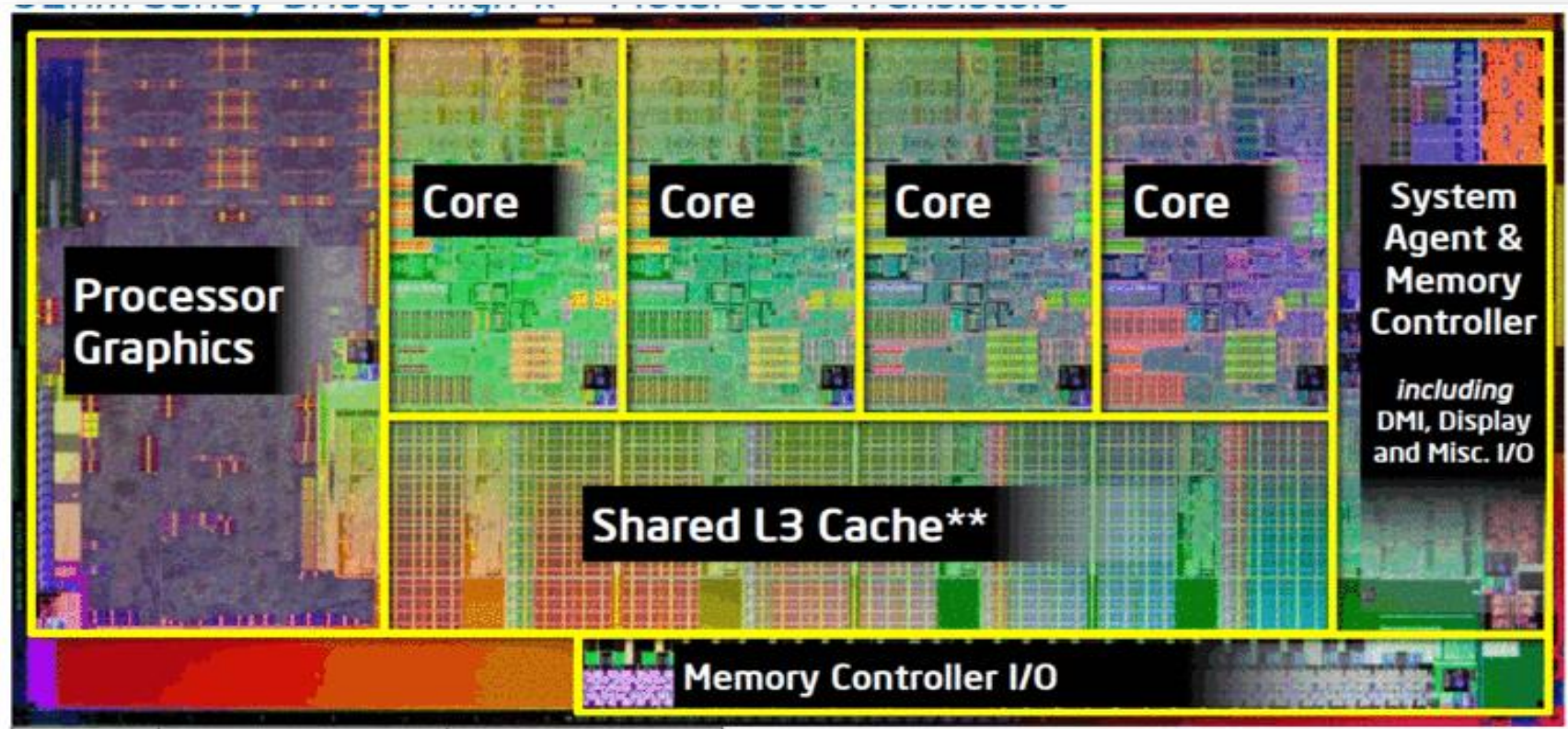


Figure 1: Current and expected eras of Intel® processor architectures

Platform 2015: Intel® Processor and Platform Evolution for the Next Decade, 2005

Intel Sandy Bridge, January 2011

- 4 to 8 core



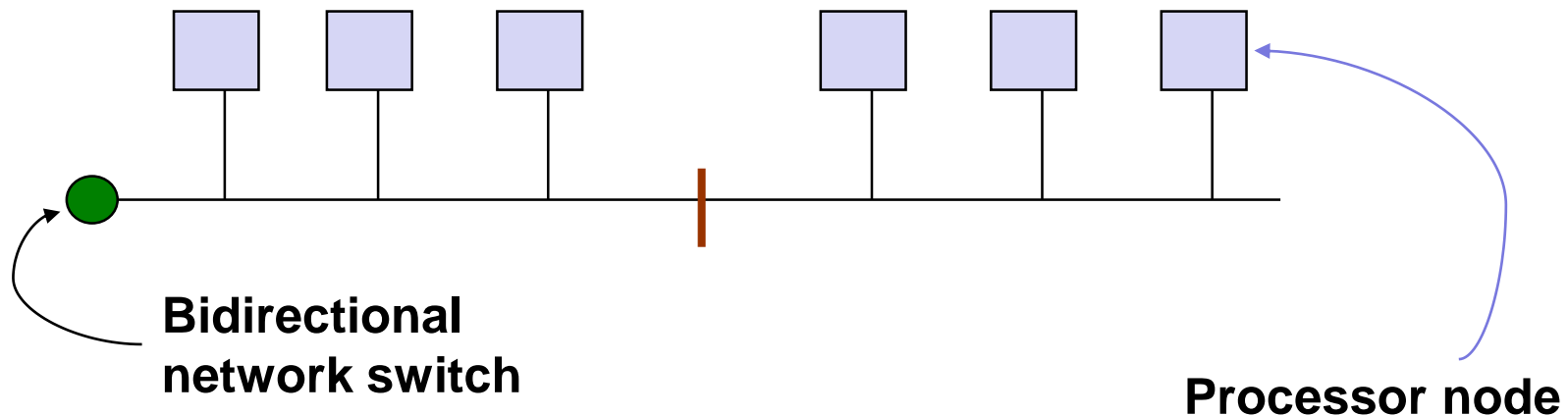
Performance Metrics of Interconnection Network

- Network cost
 - number of switches
 - number of links on a switch to connect to the network (plus one link to connect to the processor)
 - width in bits per link, length of link
- Network bandwidth (NB)
 - represents the best case
 - bandwidth of each link * number of links
- Bisection bandwidth (BB)
 - represents the worst case
 - divide the machine in two parts, each with half the nodes and sum the bandwidth of the links that cross the dividing line



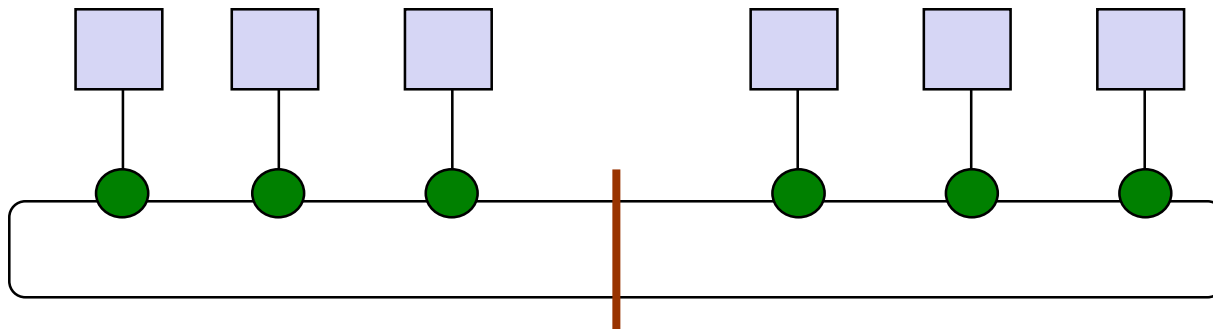
Bus Network

- N processors, 1 switch (●), 1 link (the bus)
- Only 1 simultaneous transfer at a time
 - NB (best case) = link (bus) bandwidth * 1
 - BB (worst case) = link (bus) bandwidth * 1



Ring Network

- N processors, N switches, 2 links/switch, N links
- N simultaneous transfers
 - NB (best case) = link bandwidth * N
 - BB (worst case) = link bandwidth * 2
- If a link is as fast as a bus, the ring is only twice as fast as a bus in the worst case, but is N times faster in the best case



Cell Broadband Engine (2005)

- Cell Broadband Engine (2005)
 - 8 core (SPE) + 1 core (PPE)
 - each SPE has 256KB memory
 - PS3, IBM Roadrunner(12k)



PlayStation3 の写真は PlayStation.com (Japan) から

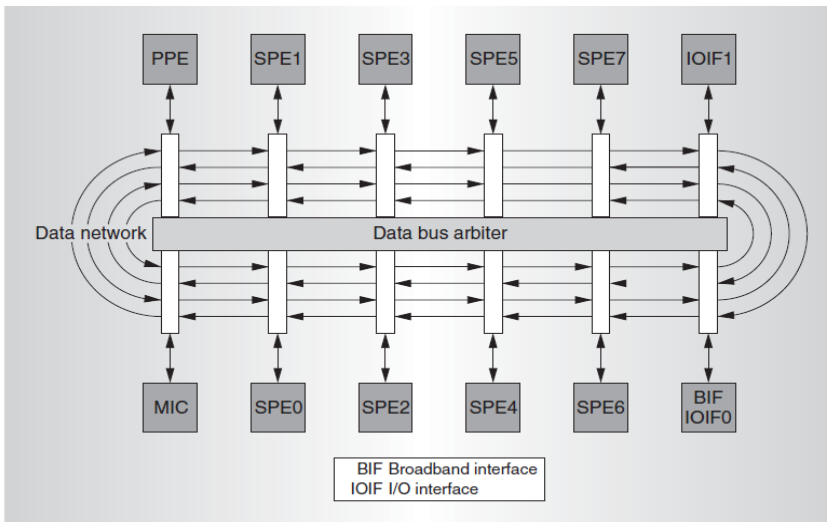


Figure 2. Element interconnect bus (EIB).

IEEE Micro, Cell Multiprocessor Communication Network: Built for Speed

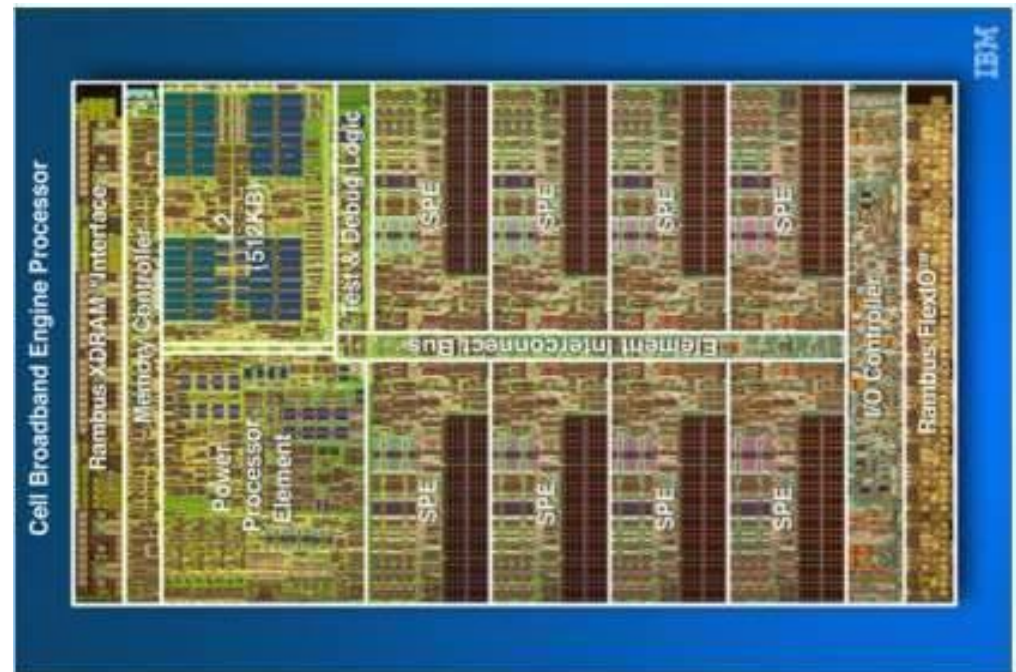
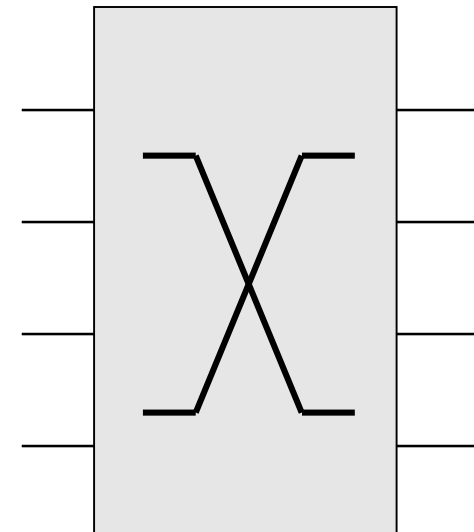
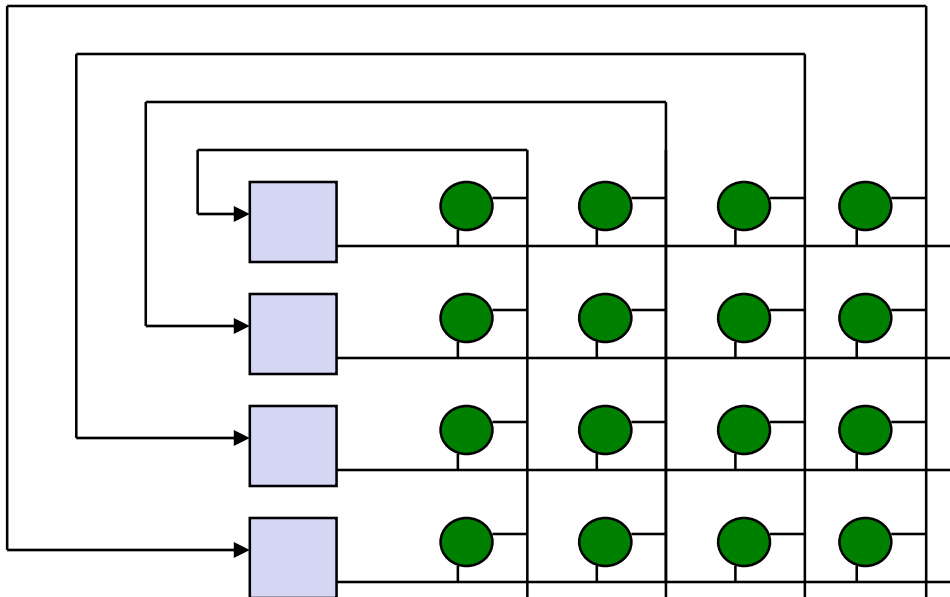


Diagram created by IBM to promote the CBEP, ©2005 from WIKIPEDIA

Crossbar (Xbar) Network

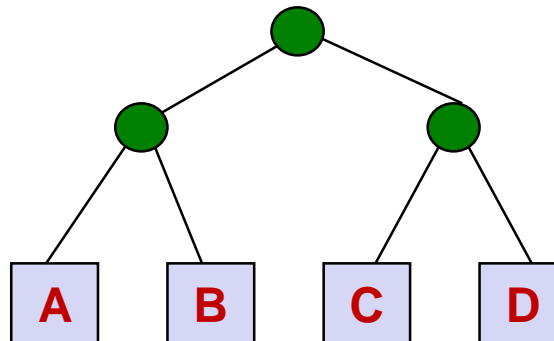
- N processors, N^2 switches (unidirectional), 2 links/switch, N^2 links
- N simultaneous transfers
 - $NB = \text{link bandwidth} * N$
 - $BB = \text{link bandwidth} * N/2$



A symbol of Xbar

Tree

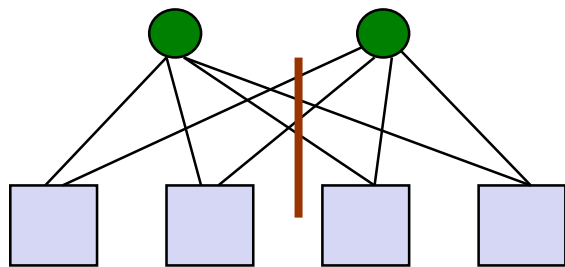
- Trees are good structures. People in CS use them all the time. **Suppose we wanted to make a tree network.**
- Any time A wants to send to C, it ties up the upper links, so that B can't send to D.
 - The bisection bandwidth on a tree is horrible - 1 link, at all times
- The solution is to 'thicken' the upper links.
 - More links as the tree gets thicker increases the bisection bandwidth



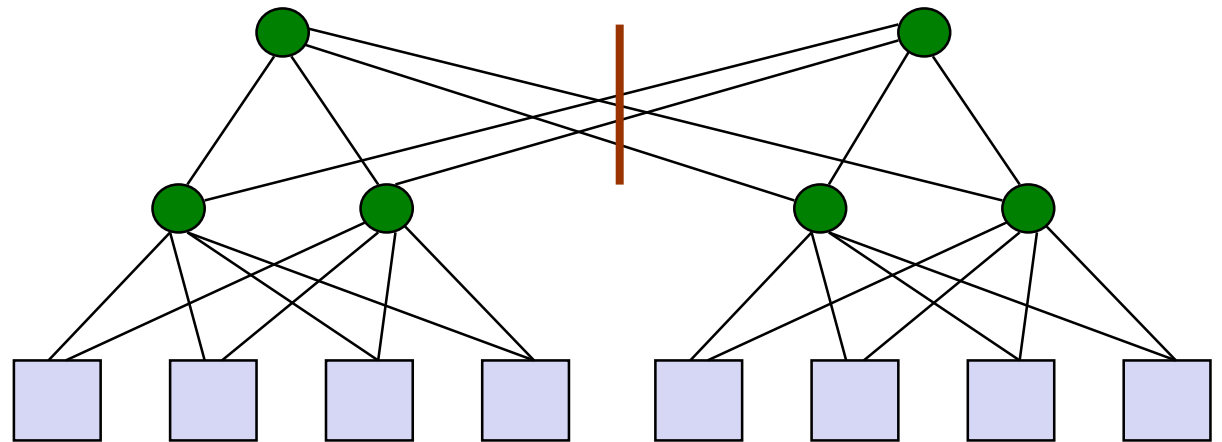
N = 4

Fat Tree

- N processors, $\log(N-1) * \log N$ switches, 2 up + 4 down = 6 links/switch, $N * \log N$ links
- N simultaneous transfers
 - $NB = \text{link bandwidth} * N \log N$
 - $BB = \text{link bandwidth} * 4$



$N = 4$

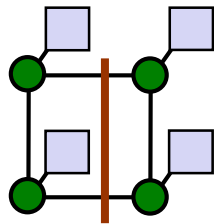


$N = 8$

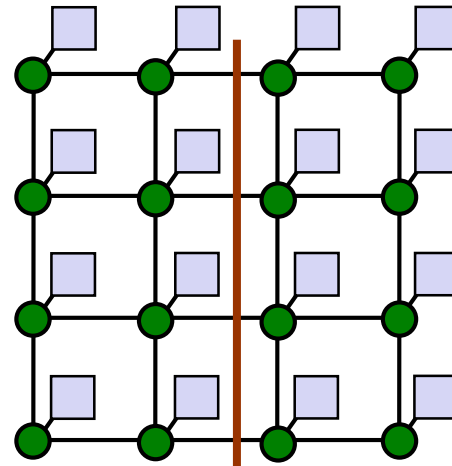


Mesh Network

- N processors, N switches, 4 links/switch, $N * (N^{1/2} - 1)$ links
- N simultaneous transfers
 - $NB = \text{link bandwidth} * 2N$
 - $BB = \text{link bandwidth} * N^{1/2}$



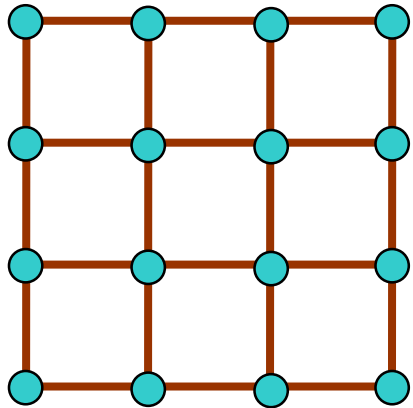
$N = 4$



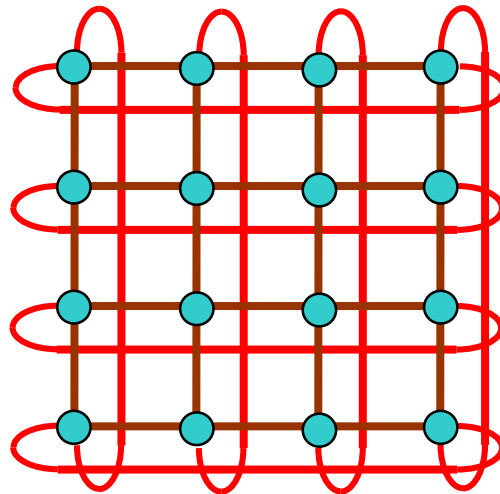
$N = 16$



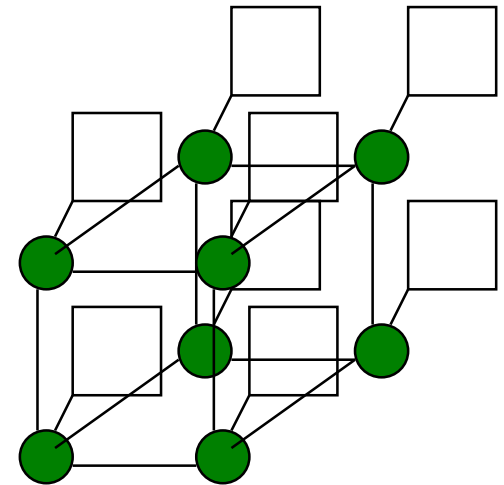
2D and 3D Mesh / Torus Network



2D Mesh



Torus

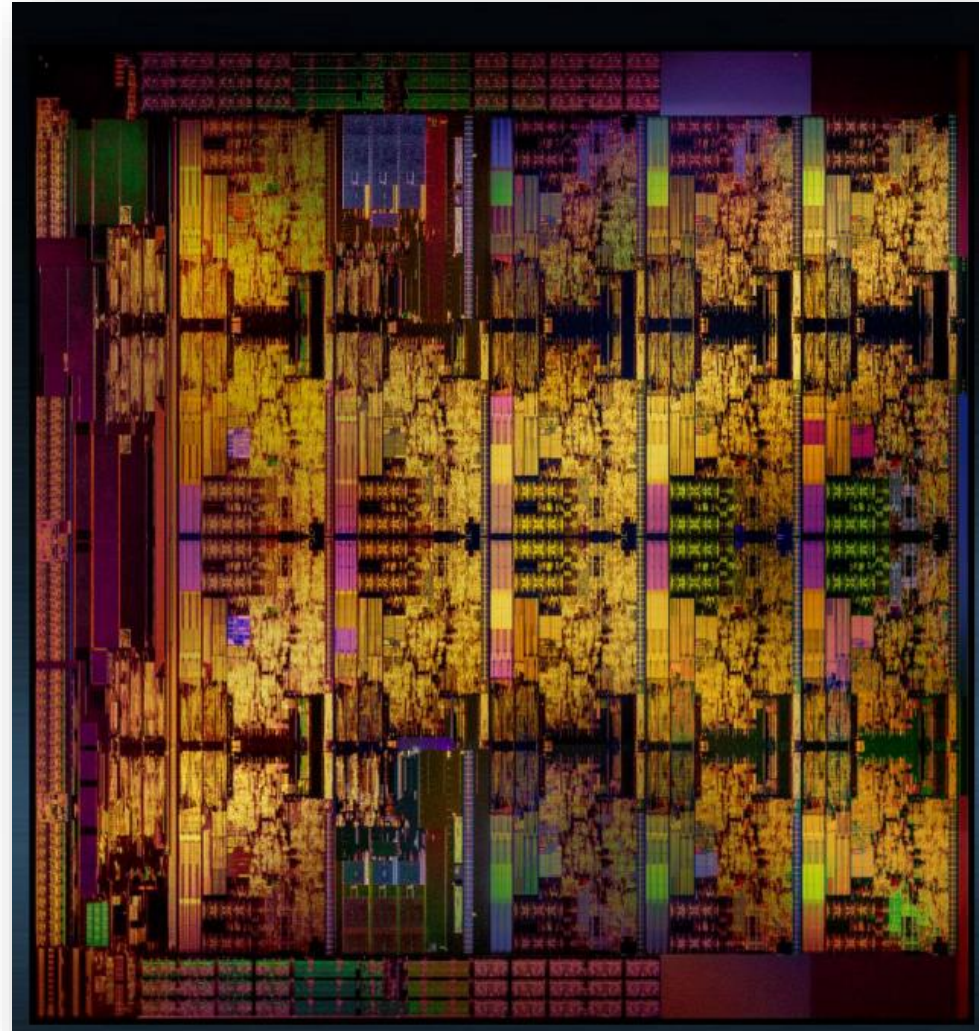


3D Mesh



Intel Skylake-X, Core i9-7980XE, 2017

- 18 core



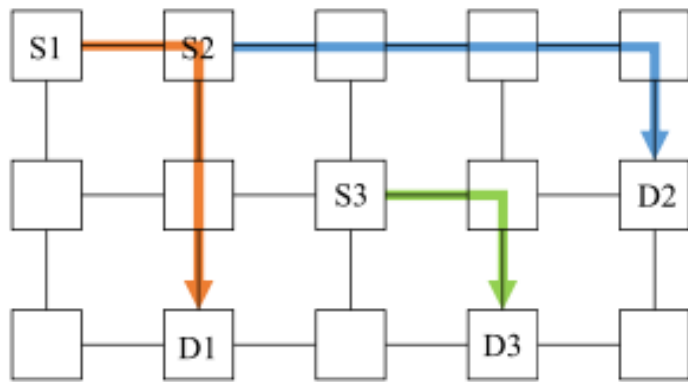
Bus vs. Networks on Chip (NoC)

- **Circuit switching**
 - a communication method where a dedicated communication path, or circuit, is established between two devices before data transmission begins
- **Packet switching**

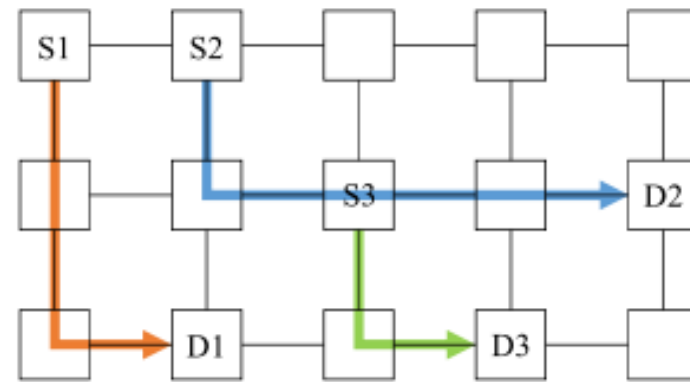


NoC and Many-core

- NoC requirements: low latency, high throughput, low cost
 - Focus on mesh topology
- Packet based data transmission via NoC routers and XY-dimension order routing



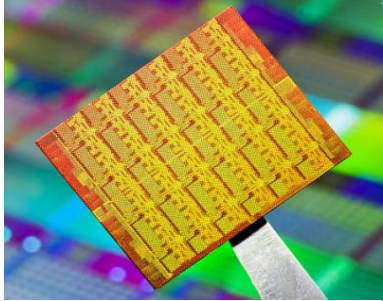
(a) XY routing



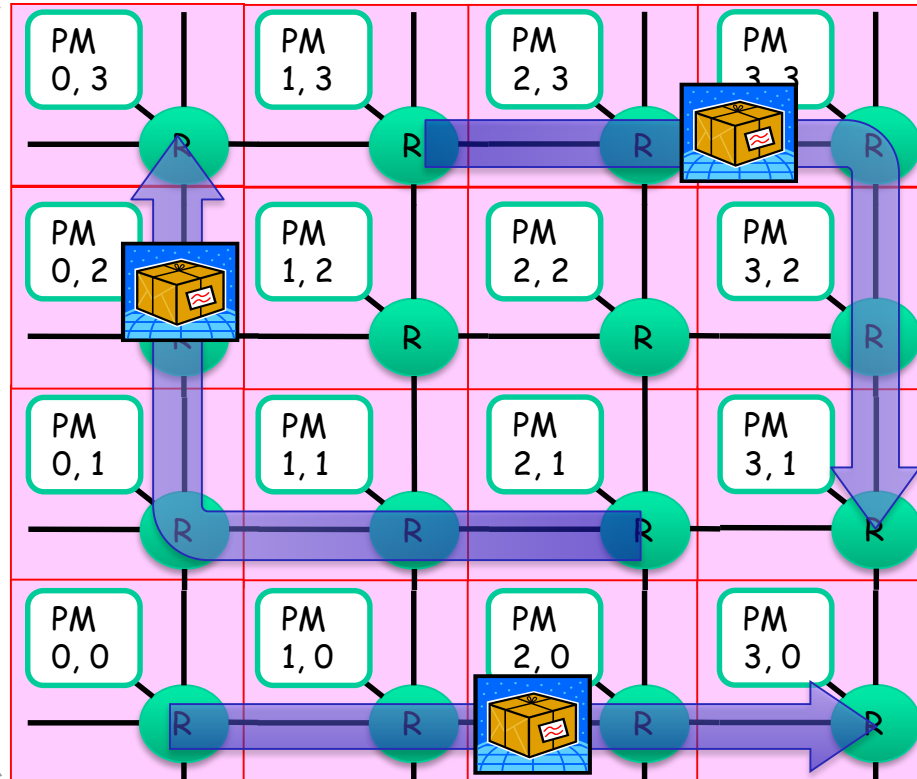
(b) YX routing



NoC and Many-core

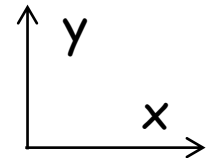


PM: Processing Module or Core
R: Router



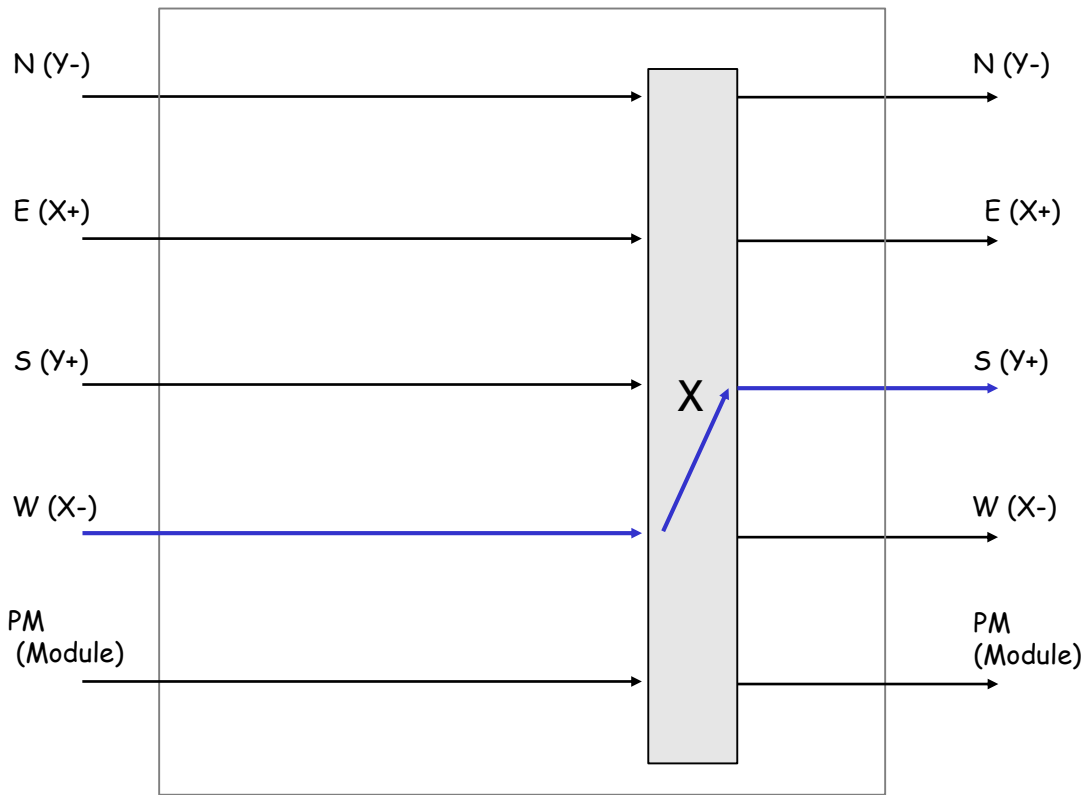
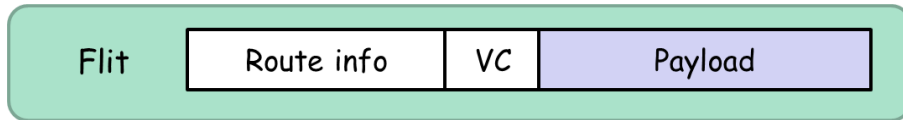
XY-dimension order routing

packet
(tag + data)

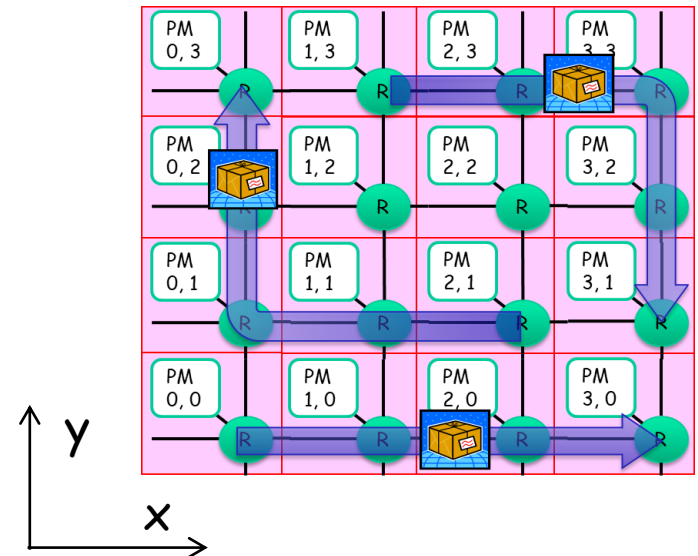
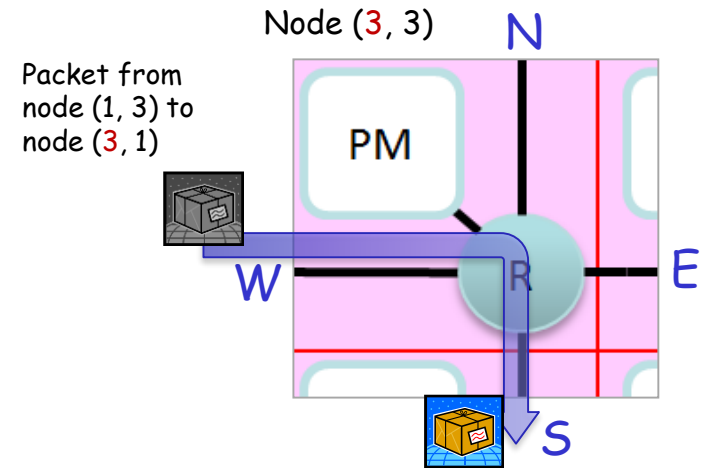


Simple NoC router architecture

- Routing computation for **XY-dimension order**

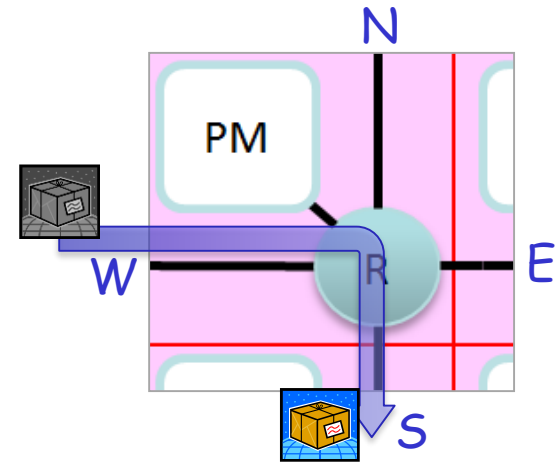
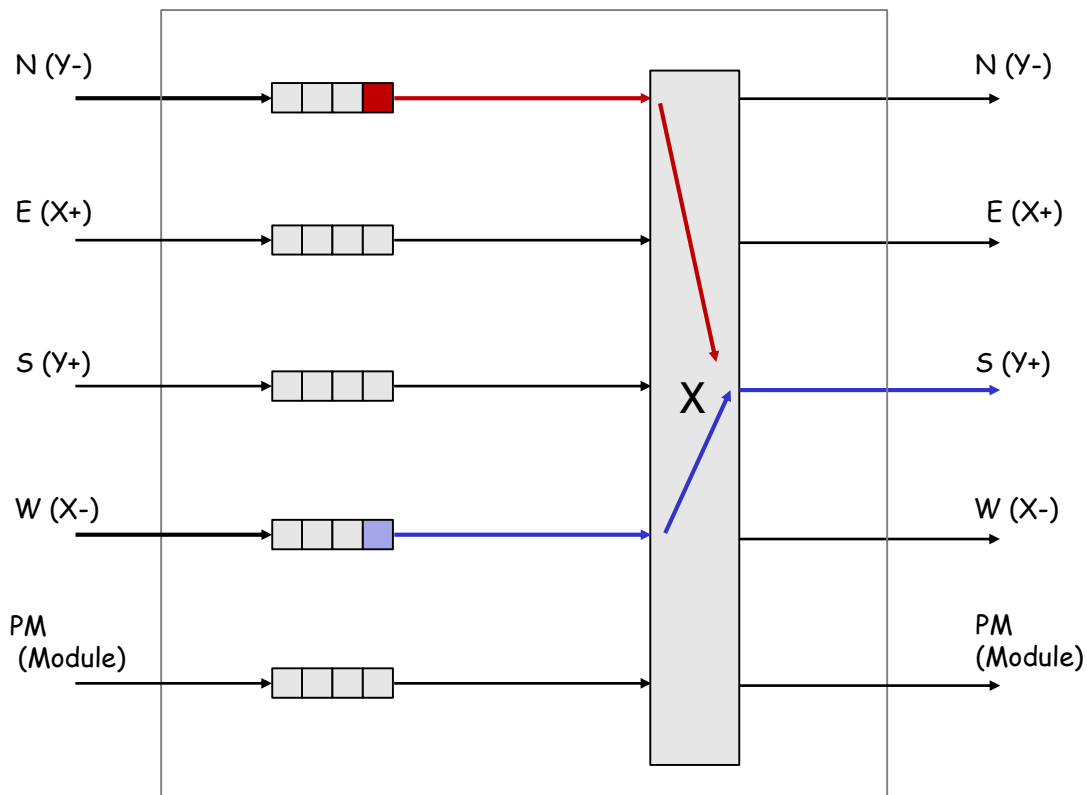


NoC router



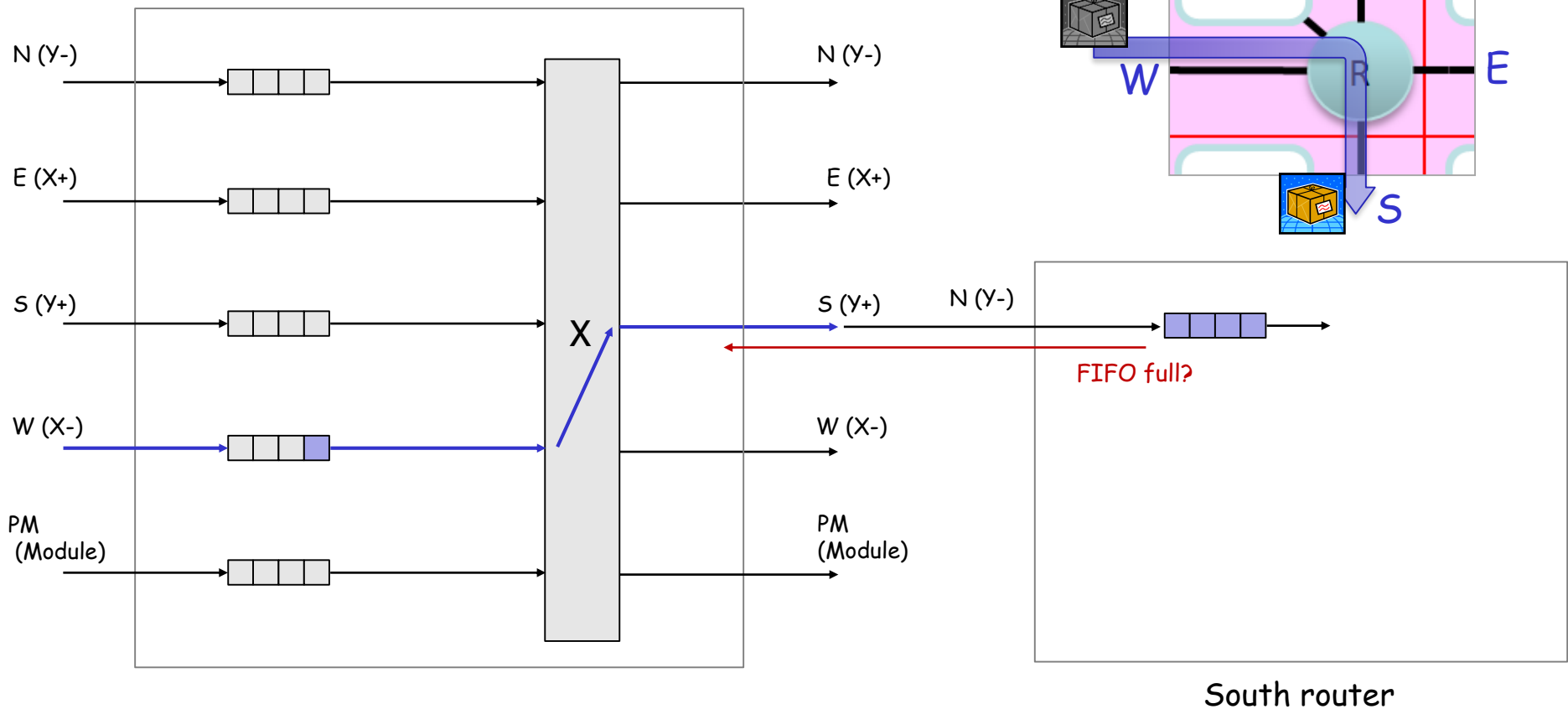
Simple NoC router architecture

- Buffering and arbitration
 - time stamp based, round robin, etc.



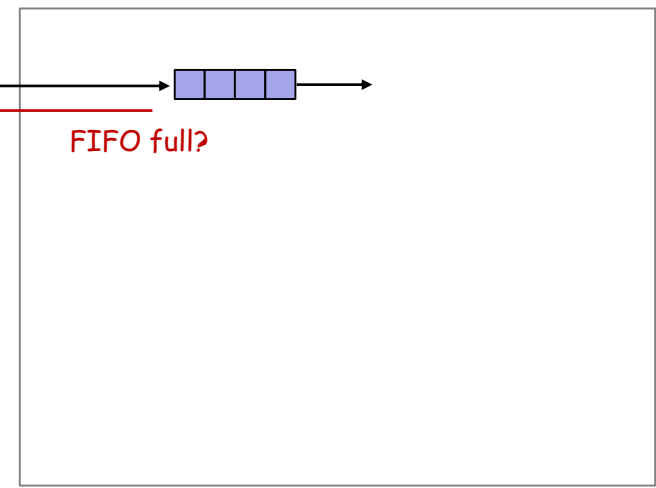
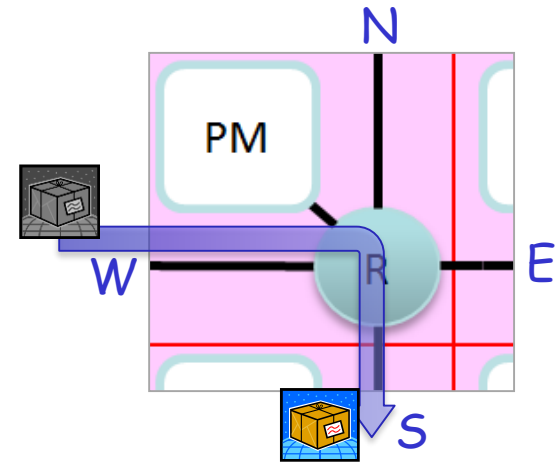
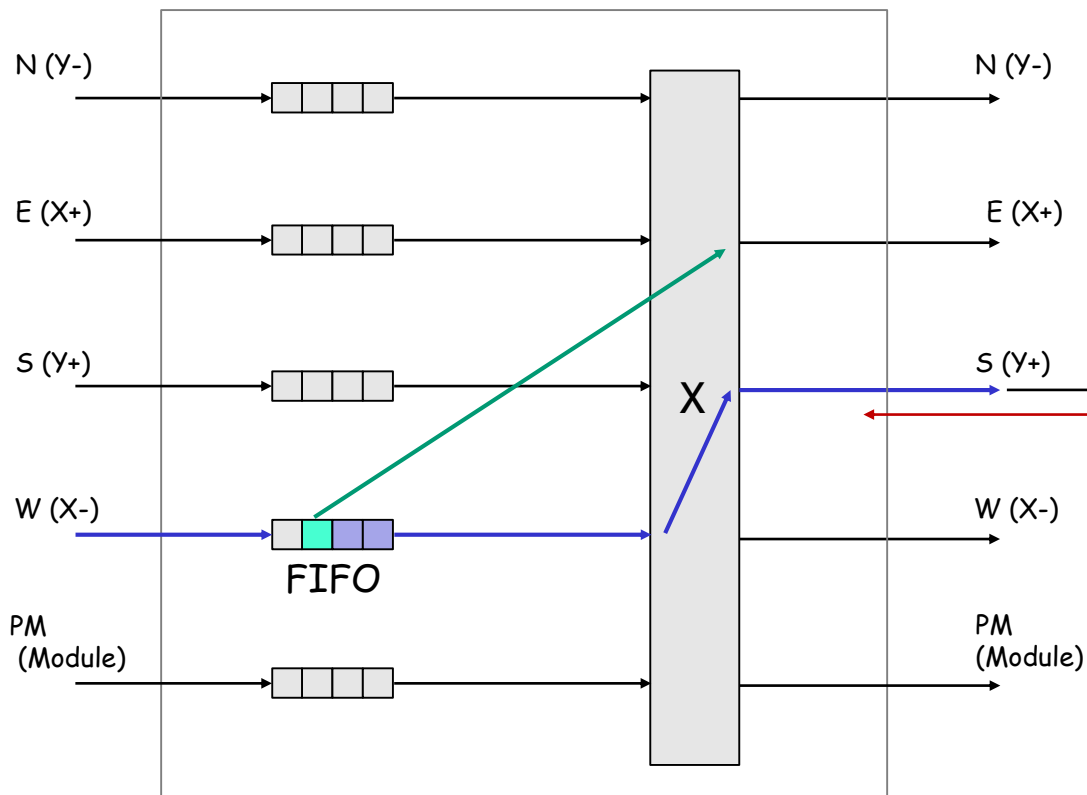
Simple NoC router architecture

- Flow control



Simple NoC router architecture

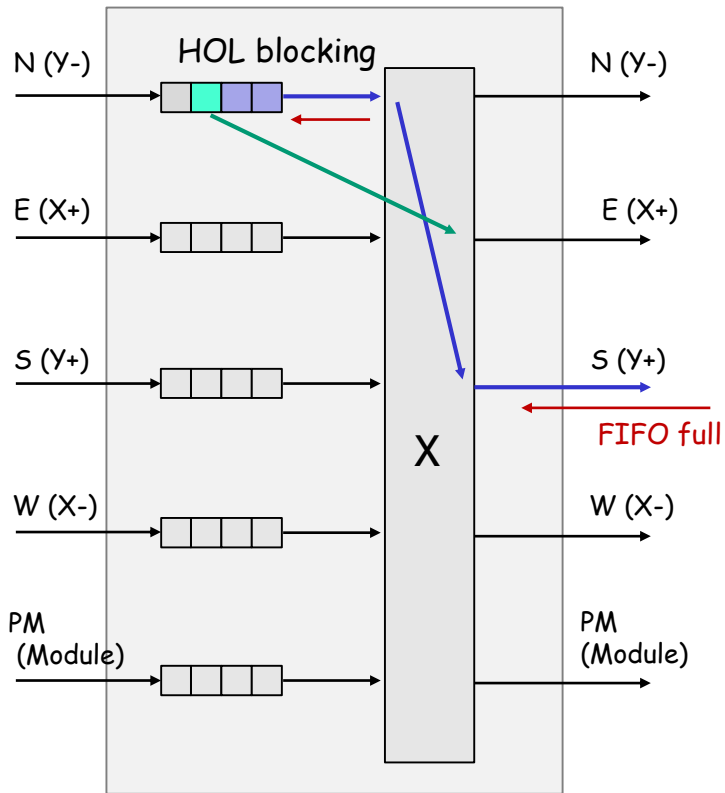
- Problem: Head-of-line (HOL) blocking



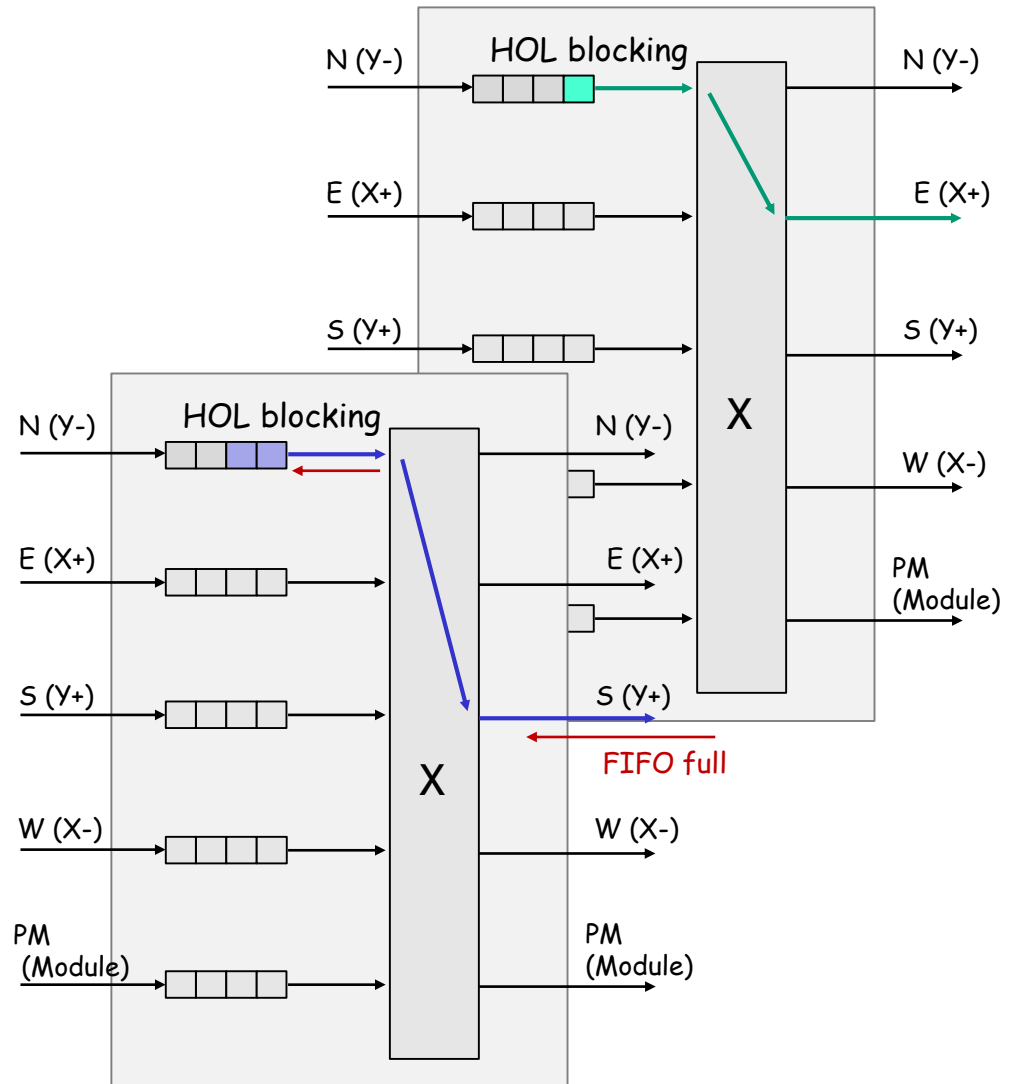
South router



Two (physical) networks to mitigate HOL ?

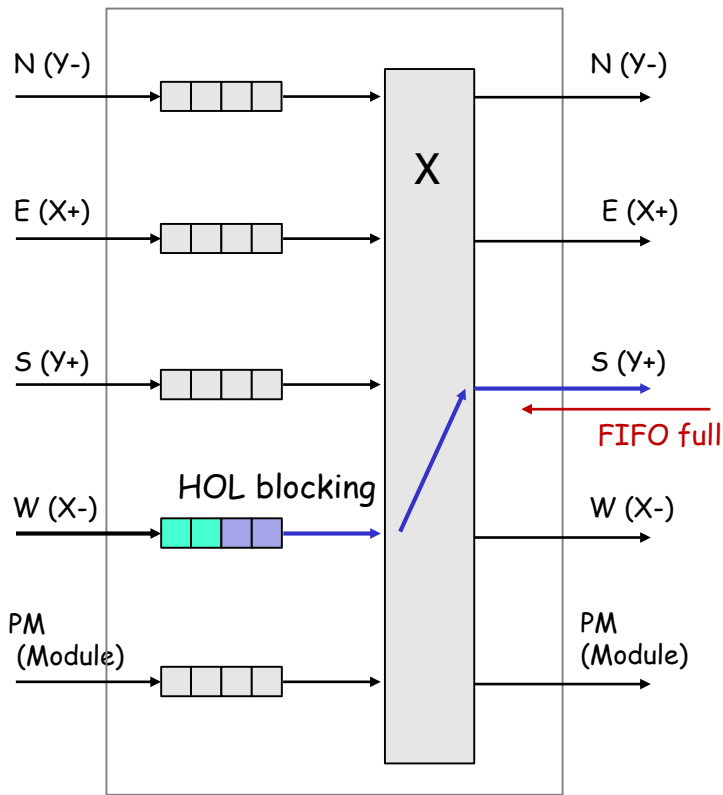
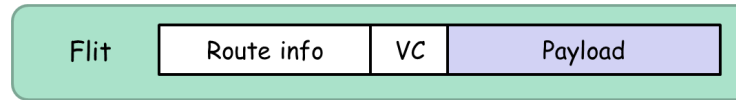


Simple NoC router

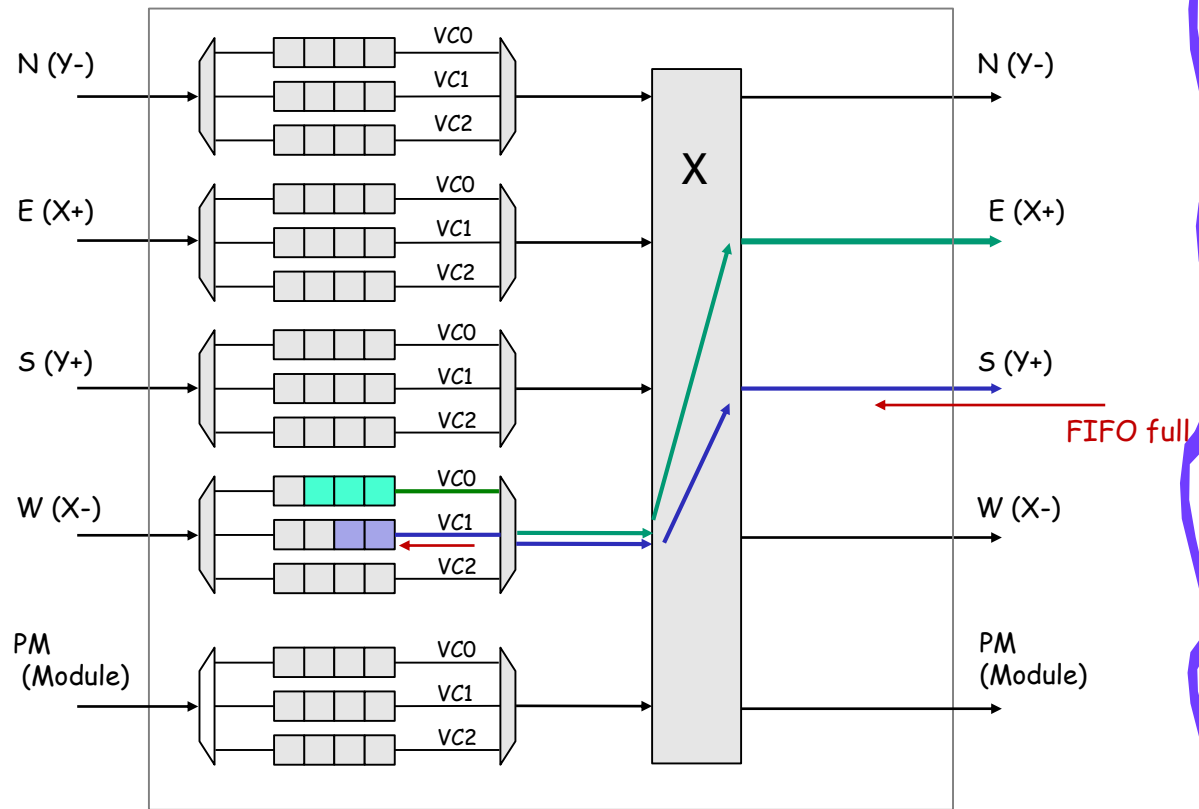


Datapath of Virtual Channel (VC) NoC router

- To mitigate head-of-line (HOL) blocking, virtual channels are used



Simple NoC router



VC NoC router

Bus vs. Networks on Chip (NoC) of mesh topology



Pipelined NoC router
Packet of multiple flits

To mitigate
head-of-line (HOL) blocking



Virtual Channel



From multi-core era to many-core era

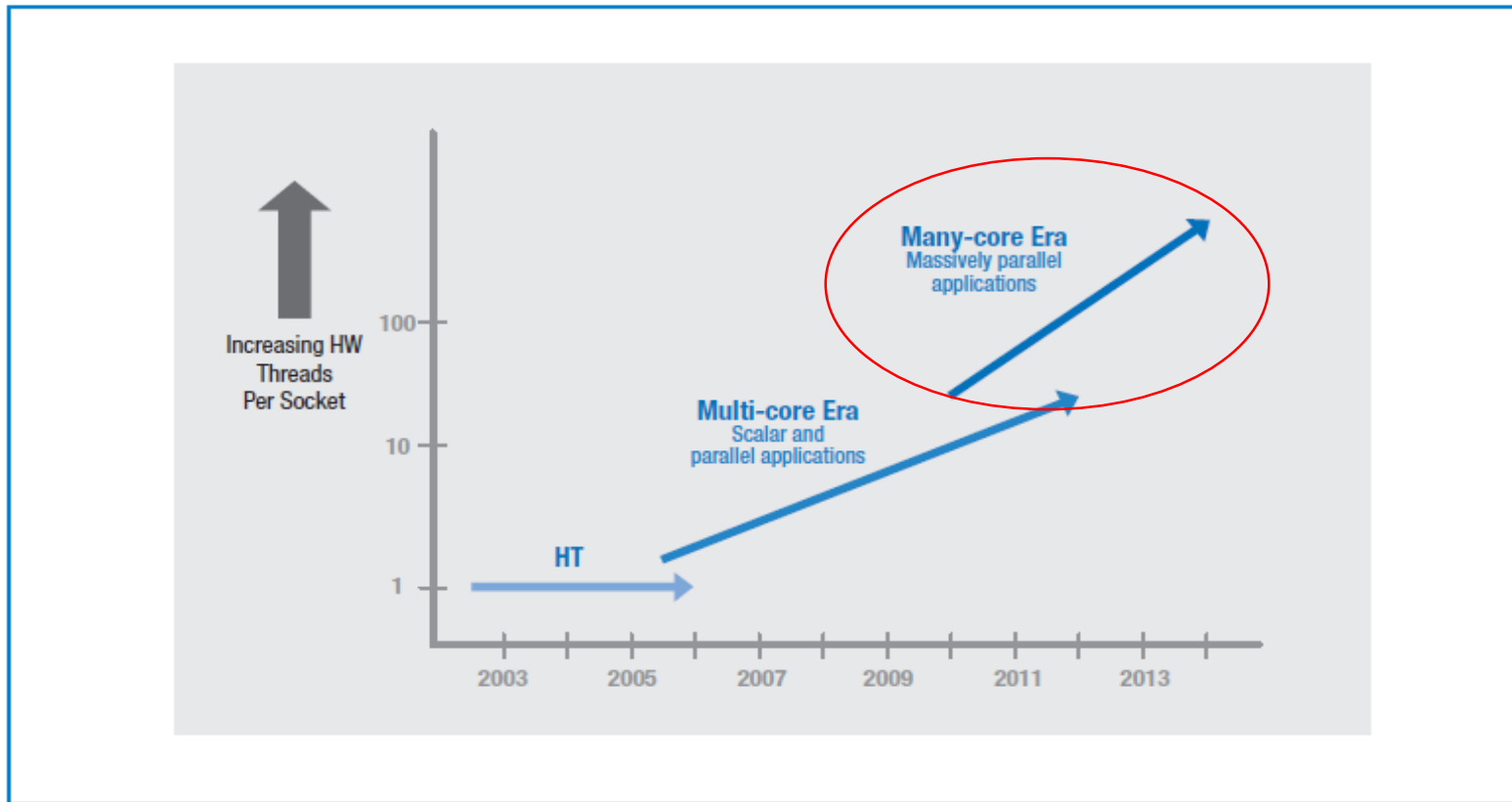


Figure 1: Current and expected eras of Intel® processor architectures

Platform 2015: Intel® Processor and Platform Evolution for the Next Decade, 2005

2021.11 Intel Alder Lake processor



2022.11 AMD EPYC 9654 processor with 96 cores



AMD EPYC™ 9004 Series Processor

All-in Feature Set support

- 12 Channels of DDR5-4800
- Up to 6TB DDR5 memory capacity
- 128 lanes PCIe® 5
- 64 lanes CXL 1.1+
- AVX-512 ISA, SMT & core frequency boost
- AMD Infinity Fabric™
- AMD Infinity Guard

Cores	AMD EPYC	Base/Boost* <small>(up to GHz)</small>	Default TDP (w)	cTDP (w)
96 cores	9654/P	2.40/3.70	360w	320-400w
84 cores	9634	2.25/3.70	290w	240-300w
64 cores	9554/P	3.10/3.75	360w	320-400w
64 cores	9534	2.45/3.70	280w	240-300w
48 cores	→ 9474F	3.60/4.10	360w	320-400w
	9454/P	2.75/3.80	290w	240-300w
32 cores	→ 9374F	3.85/4.30	320w	320-400w
32 cores	9354/P	3.25/3.80	280w	240-300w
32 cores	9334	2.70/3.90	210w	200-240w
24 cores	→ 9274F	4.05/4.30	320w	320-400w
	9254	2.90/4.15	200w	200-240w
16 cores	→ 9174F	4.10/4.40	320w	320-400w
	9124	3.00/3.70	200w	200-240w



The Free Lunch Is Over

- Tuning, Optimization, and Parallel processing (Concurrency)

Free Lunch

Programmers haven't really had to worry much about performance or concurrency because of Moore's Law

Why we did not see 4GHz processors in Market?

The traditional approach to application performance was to simply wait for the next generation of processor; most software developers did not need to invest in performance tuning, and enjoyed a "free lunch" from hardware improvements.

The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software by Herb Sutter, 2005

Distributed Memory Multi-Processor Architecture

A PC cluster or parallel computers for higher performance

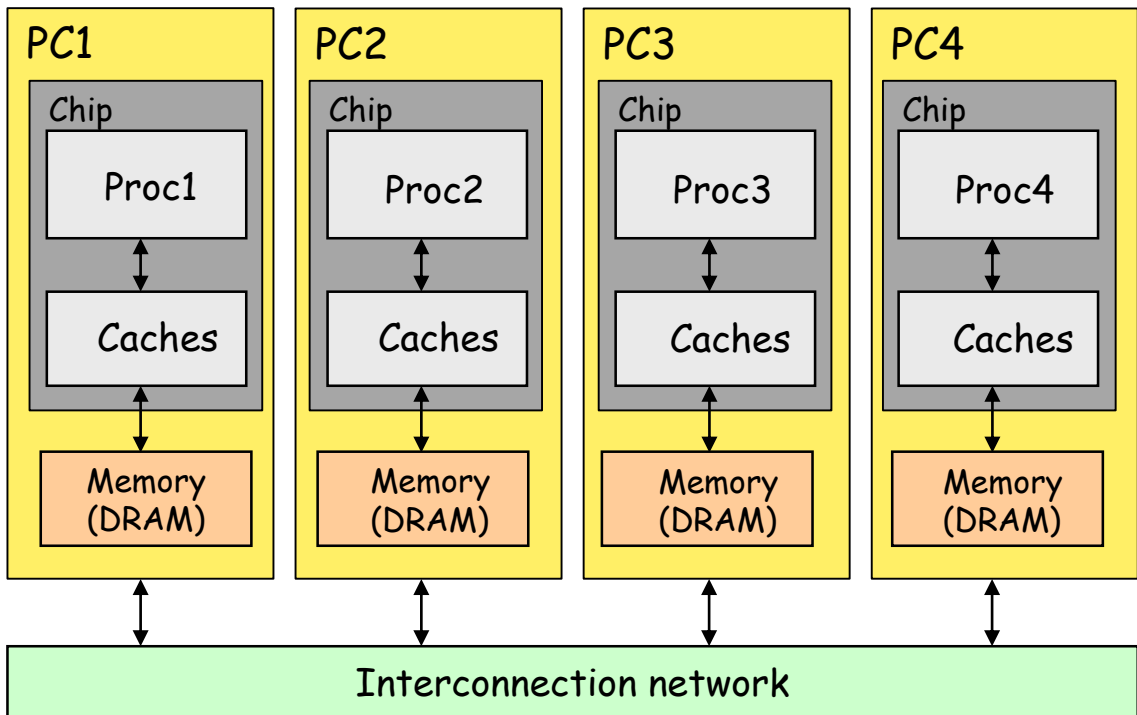
Each memory module is associated with a processor

Using explicit send and receive functions (message passing) to obtain the data required.

Who will send and receive data? How?



PC cluster



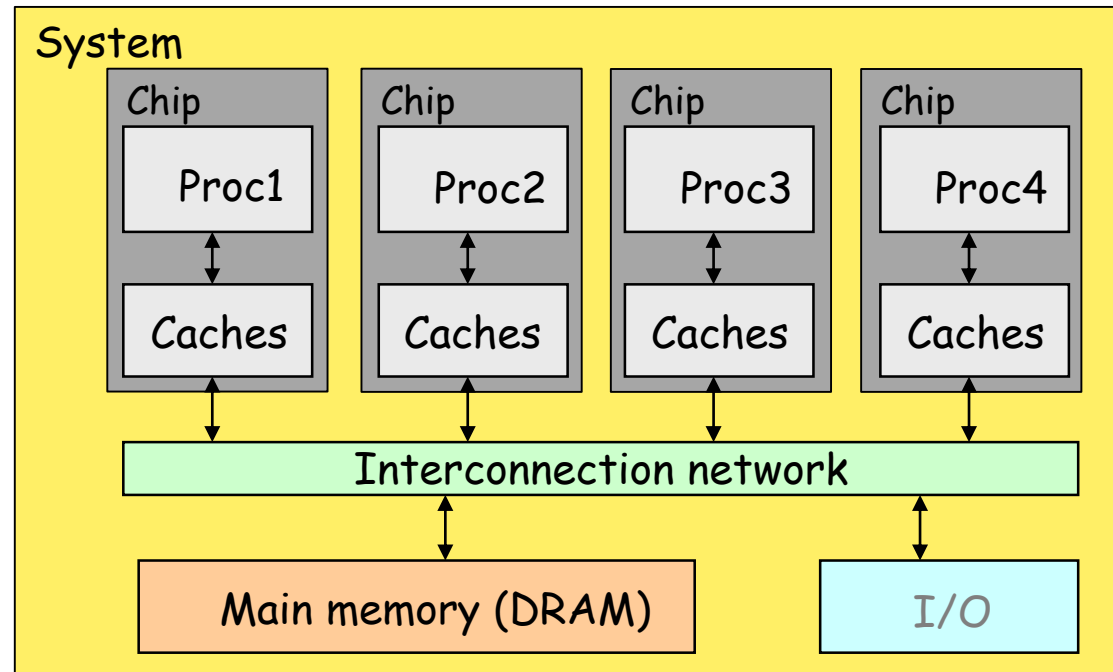
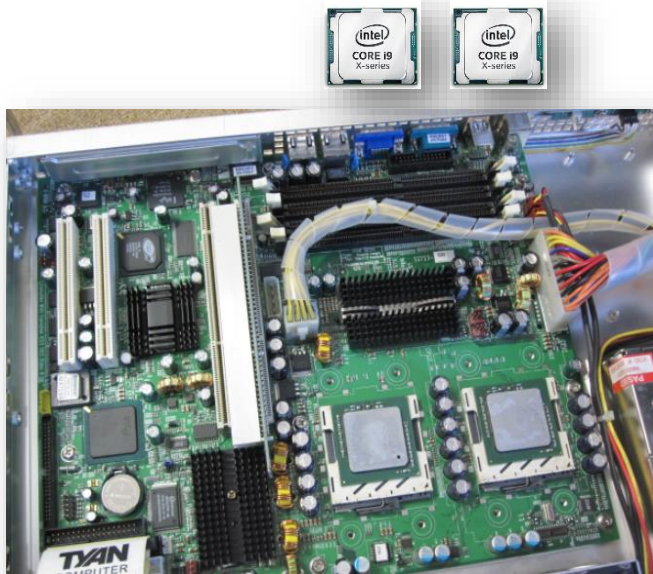
Shared Memory Multi-Processor Architecture

All the processors can access the same address space of the main memory (shared memory) through an interconnection network.

The shared memory or **shared address space (SAS)** is used as a means for communication between the processors.

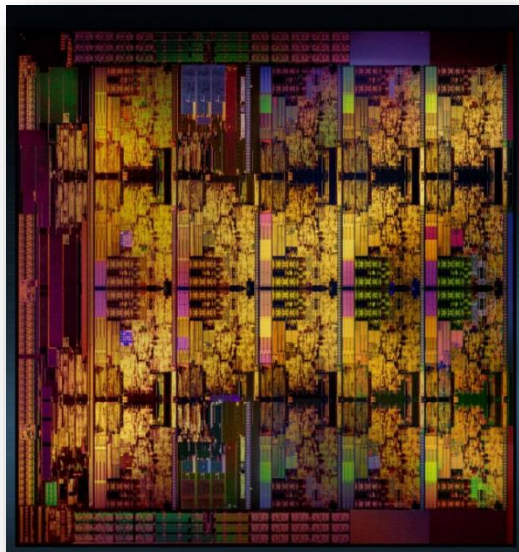
What are the means to obtain the shared data?

What are the advantages of shared memory?

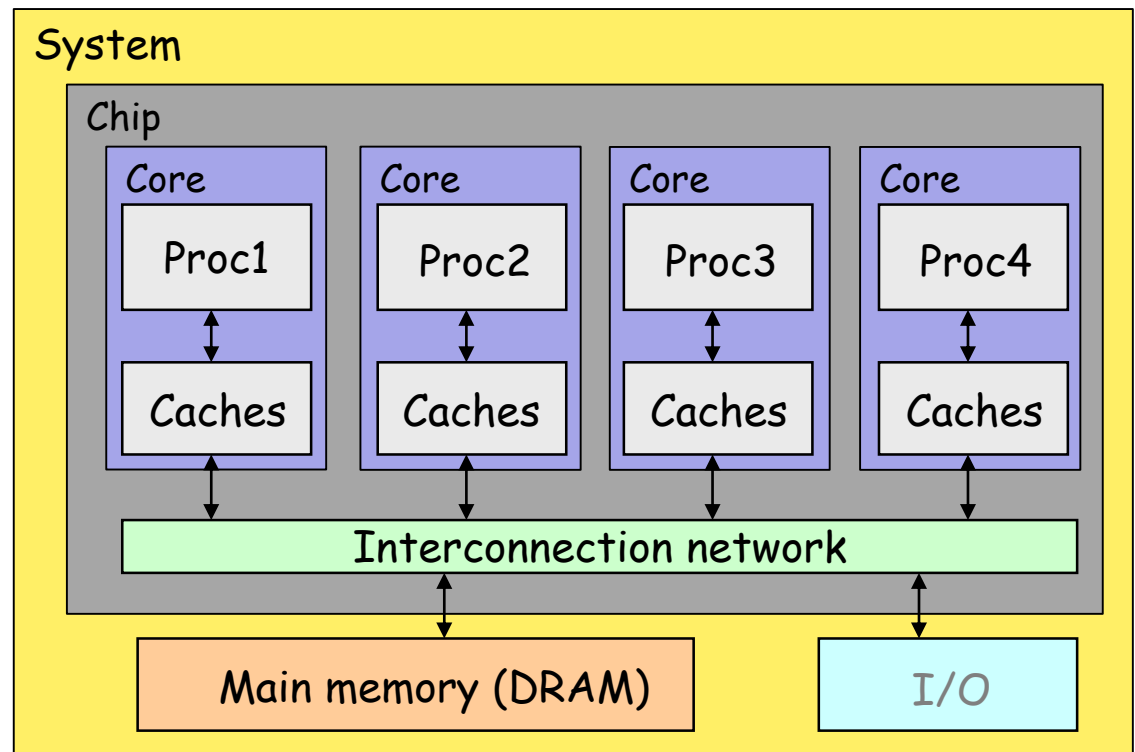


Shared memory many-core architecture

The single-chip integrates many cores (conventional processors) and an interconnection network.

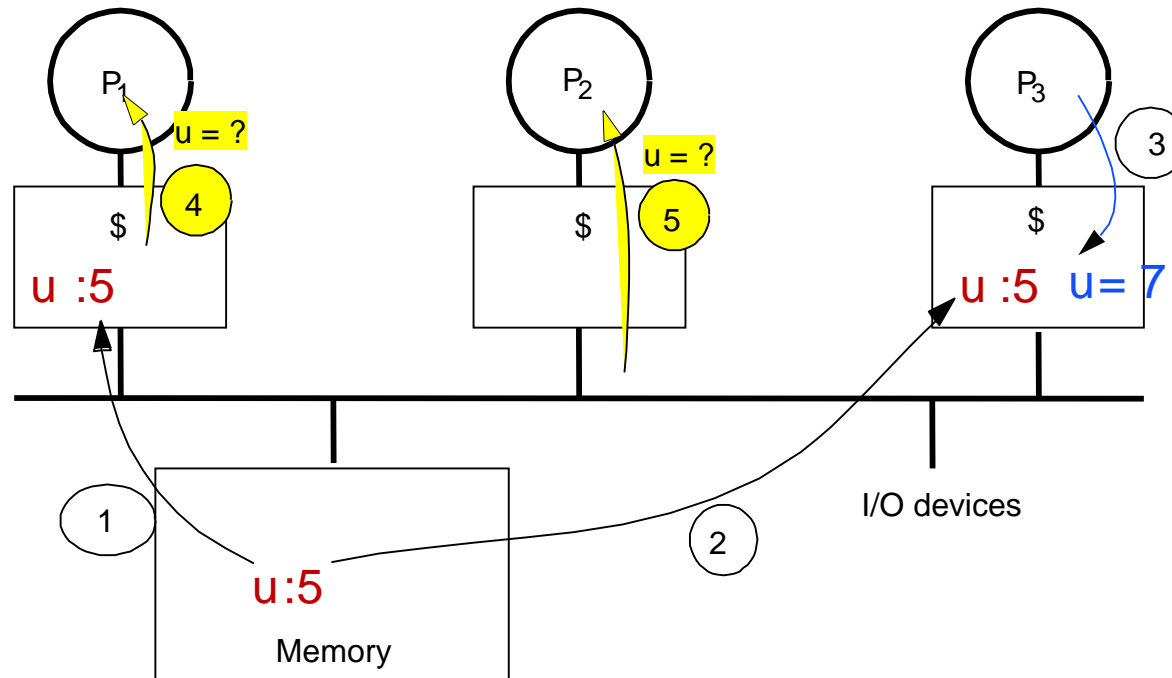


Intel Skylake-X, Core i9-7980XE, 2017



Cache Coherence Problem

- Processors see different values for shared data u after event 3
- With **write-back caches**, value written back to memory depends on which cache flushes or writes back value when
 - Processes accessing main memory may see stale (out-of-date) value
- Unacceptable for programming, and its frequent!



Cache coherence and enforcing coherence



- **Cache coherence**
 - All reads by any processor must return the most recently written value
 - Writes to **the same location** by any two processors are seen in the same order by all processors
- **Cache coherence protocols**
 - **Snooping (write invalidate / write update)**
 - Each core tracks sharing status of each block
 - **Directory based**
 - Sharing status of each block kept in one location



Memory consistency: problem in multi-core context

- Assume that $A=0$ and $\text{Flag}=0$ initially
- Core 1 (C1) writes data into A and sets Flag to tell C2 that data value can be read (loaded) from A .
- C2 waits till Flag is set and then reads (loads) data from A .
- What is the printed value by C2?

C1 (Core 1)

```
A = 3;  
Flag = 1;
```

C2 (Core 2)

```
while (Flag==0);  
print A;
```



Problem in multi-core context

- If the two writes (stores) of different addresses on *C1* can be **reordered**, it is possible for *C2* to read 0 from variable *A*.
- This can happen on most modern processors.
 - For single-core processor, **Code1** and **Code2** are equivalent. These writes may be **reordered** by compilers statically or by OoO execution units dynamically.

Code1

```
A = 3;  
Flag = 1;
```

Code2

```
Flag = 1;  
A = 3;
```

C1 (Core 1)

```
Flag = 1;  
  
A = 3;
```

C2 (Core 2)

```
while (Flag==0);  
print A;
```

The printed value by *C2* will be 0 or 3.

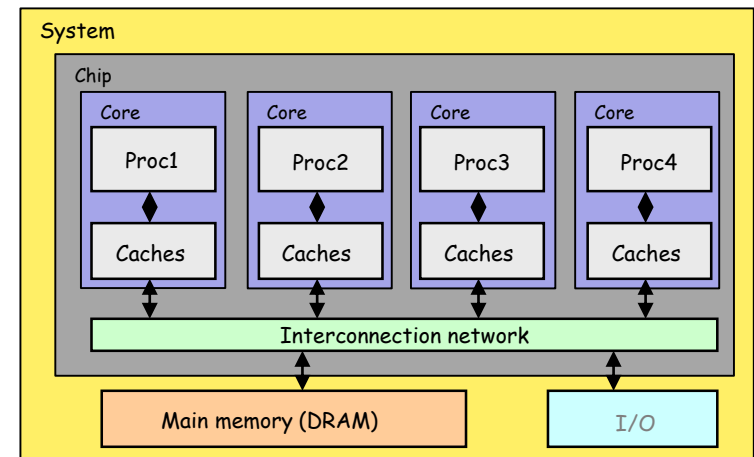
Memory Consistency Models

- A single-core processor can reorder instructions subject only to **control and data dependence constraints**
- These constraints are not sufficient in **shared-memory multi-cores**
 - simple parallel programs may produce counter-intuitive results
- **Question:** what **constraints** must we put on single-core instruction reordering so that
 - shared-memory programming is intuitive
 - but we do not lose single-core performance?
- The answers are called **memory consistency models** supported by the processor
 - Memory consistency models are all about **ordering constraints** on independent memory operations **in a single-core's instruction stream**



Key components of many-core processors

- **Interconnection network**
 - connecting many modules on a chip achieving high throughput and low latency with **NoC routers**
- **Main memory and caches**
 - Caches are used to reduce latency and to lower network traffic
 - A parallel program has private data and shared data
 - New issues are **cache coherence** and **memory consistency**
- **Core**
 - High-performance superscalar processor providing a hardware mechanism to support thread synchronization



Computer Architecture & Design

