Guest Lecturer

- ***Dr. Matthias Becker***
  - Assitant Professor (Tenure Track) at KTH Royal Institute of Technology
  - Expert in ***many-core architectures***
  - B.Eng. degree in Mechatronics/Automation Systems from the University of Applied Sciences Esslingen, Germany in 2011
  - M.Sc. degree in Computer Science specializing in Embedded Computing from the University of Applied Sciences Munich, Germany in 2013
  - In 2015 and 2017, Licentiate and PhD degree in Computer Science and Engineering from Mälardalen University, respectively

https://www.kth.se/profile/mabecker?l=en

Learning Objectives of this Lecture

After this lecture you should be able to explain and reflect on

- Why to use these platforms
- Main differences of heterogeneous and many-core platforms
- What are their benefits?
- What are their challenges?
- Workload types and suitable HW platforms
- Execution Paradigms

Outline

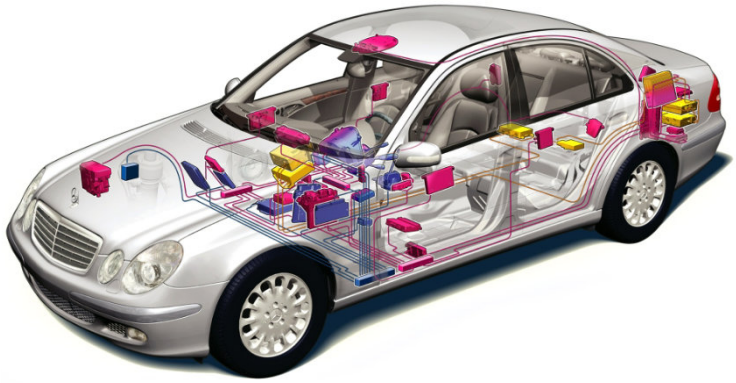
- Definition and main concepts
- Typical workloads and requirements
- Heterogeneous processor / Many-core processor
  - Interconnect
  - Arrangement of memory and processing units
  - Challenges for predictability
  - Scheduling and execution paradigms
  - Industrial and academic examples
- Conclusions

Note: There are many slides but only because of animations!


Motivation?

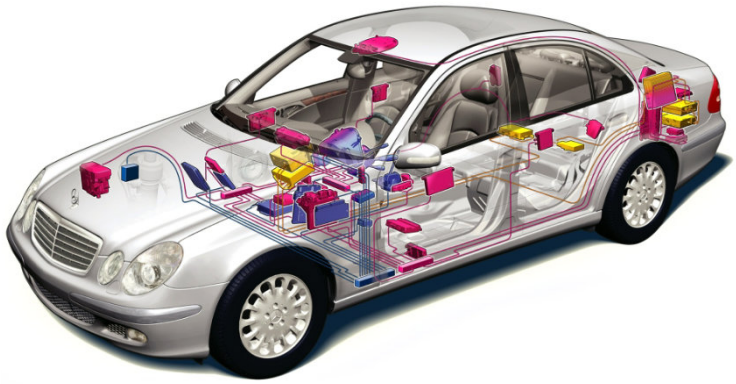
Motivation and Background

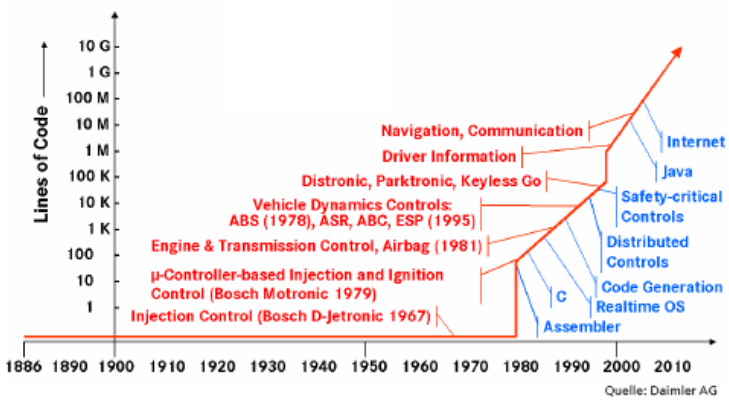
- Software complexity in many areas increases
  - Improved functionality
  - New functionality



Motivation and Background

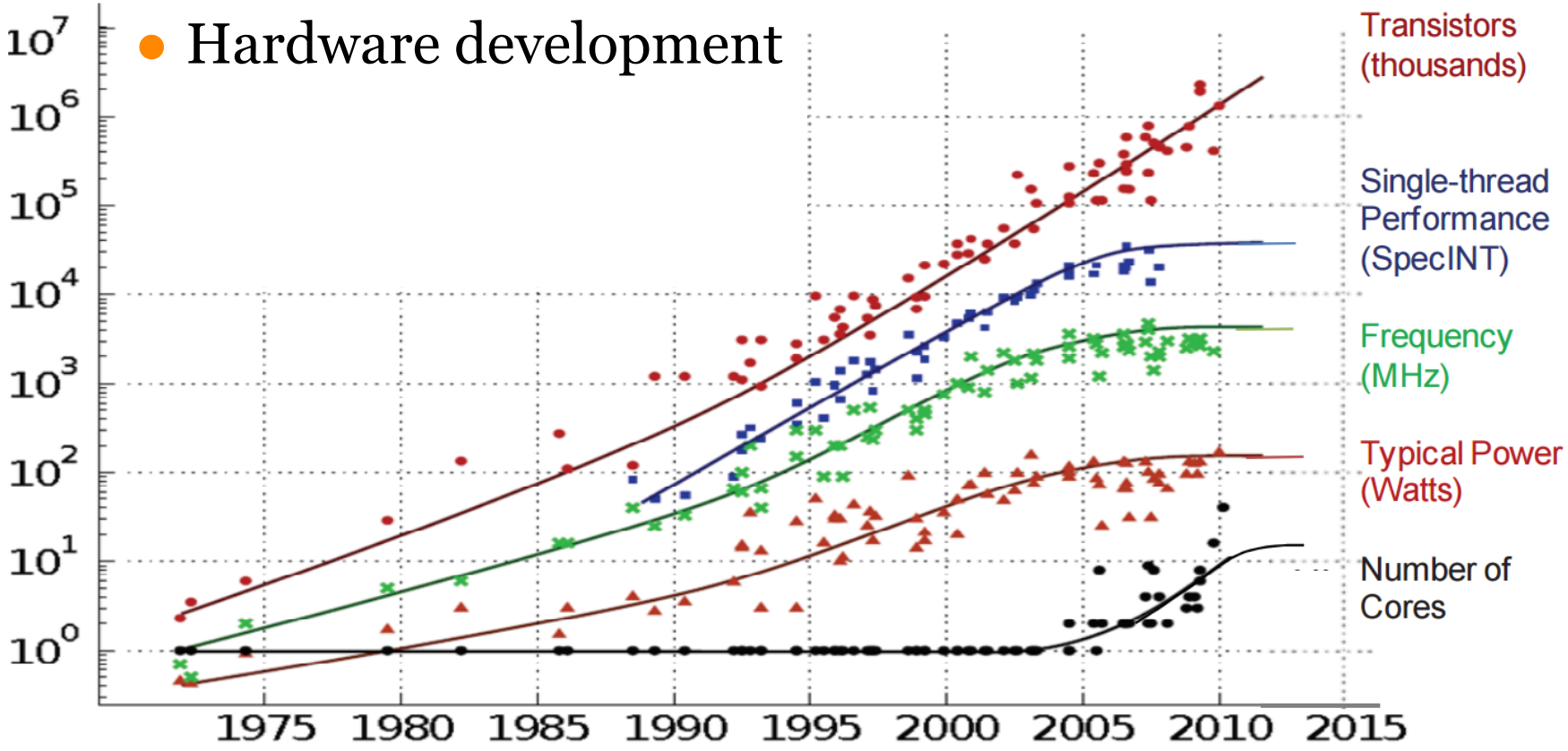
- Software complexity in many areas increases
  - Improved functionality
  - New functionality





Motivation and Background

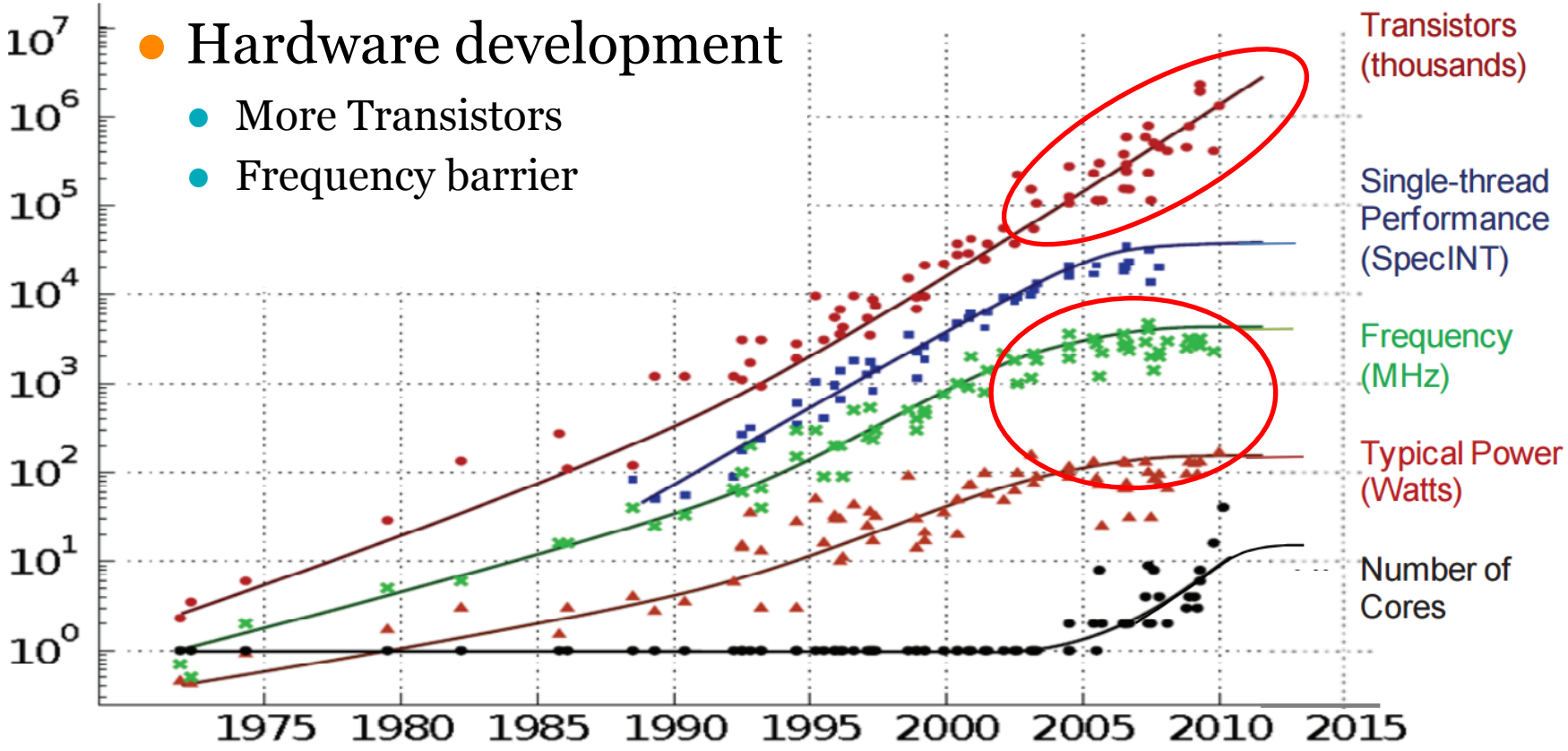
- Hardware development



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten

Motivation and Background

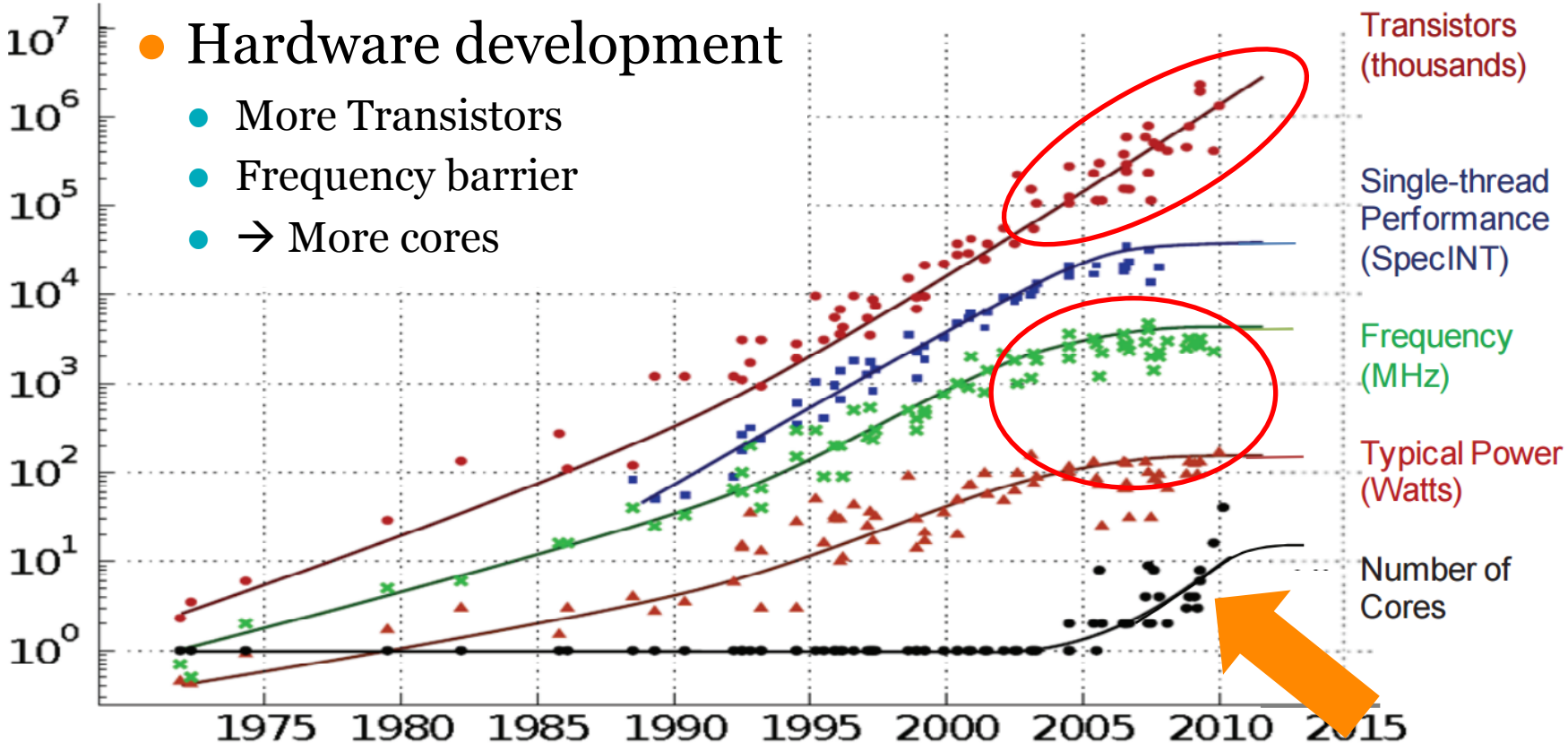
- Hardware development
  - More Transistors
  - Frequency barrier



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten

Motivation and Background

- Hardware development
  - More Transistors
  - Frequency barrier
  - → More cores

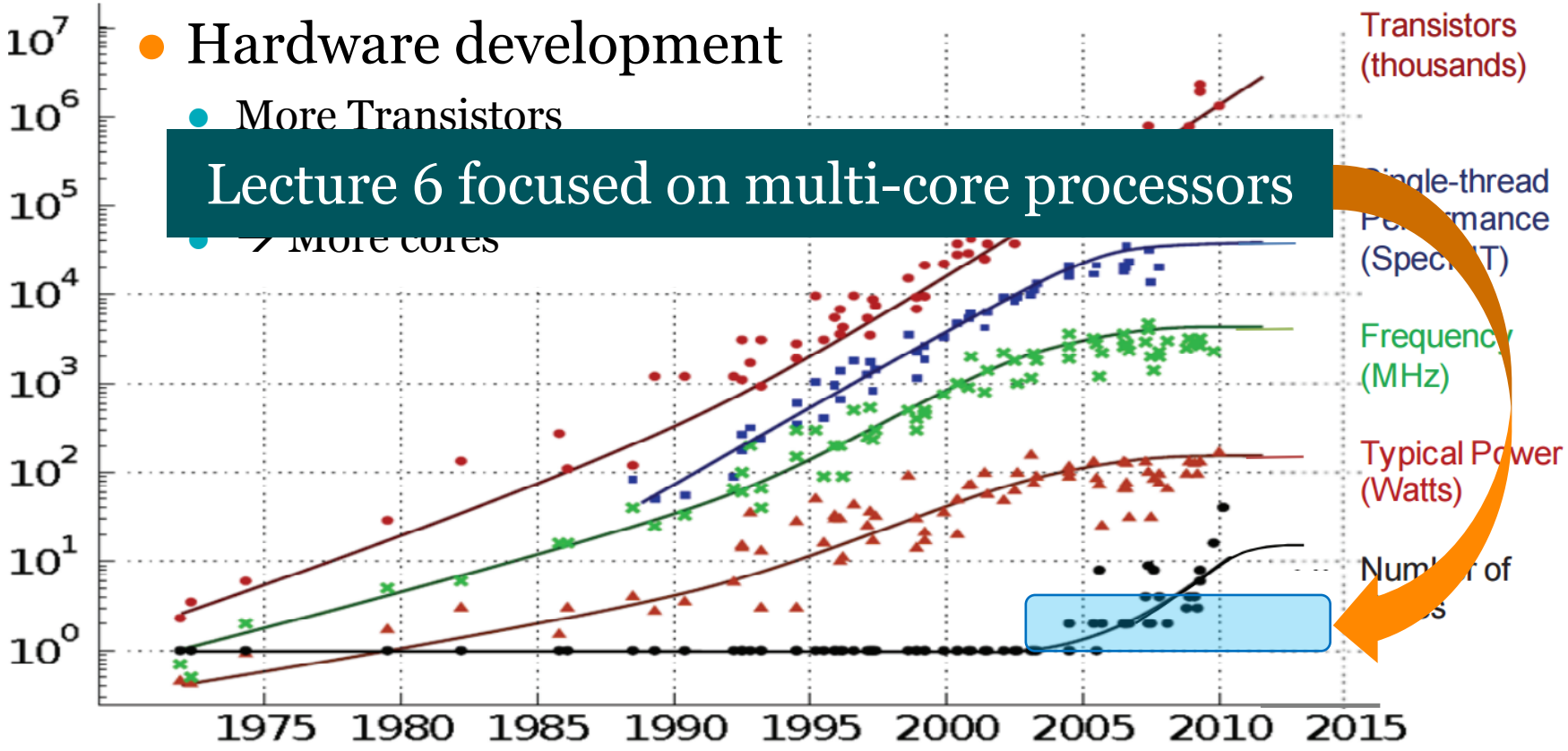


Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten

Motivation and Background

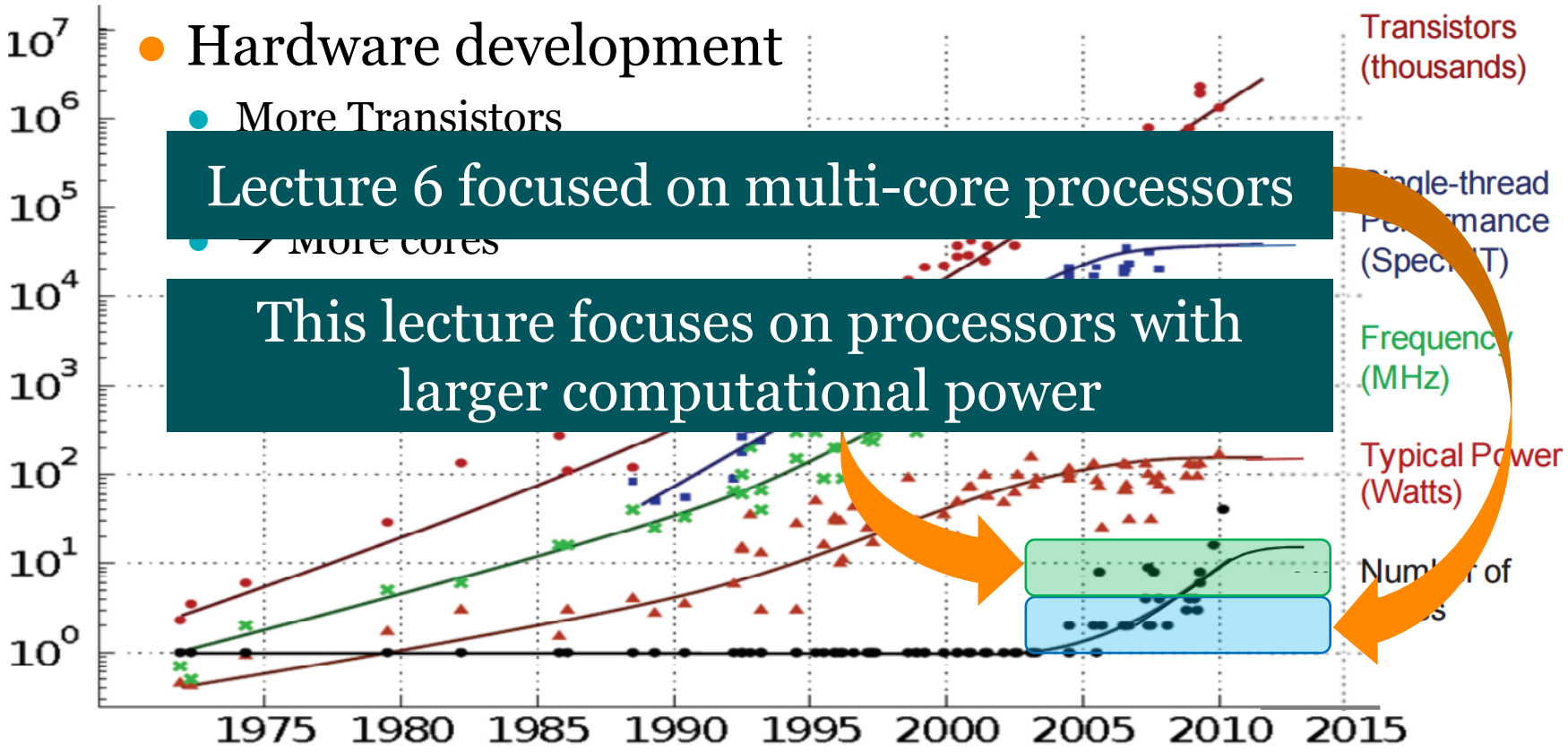
- Hardware development
  - More Transistors
  - → More cores

Lecture 6 focused on multi-core processors



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten

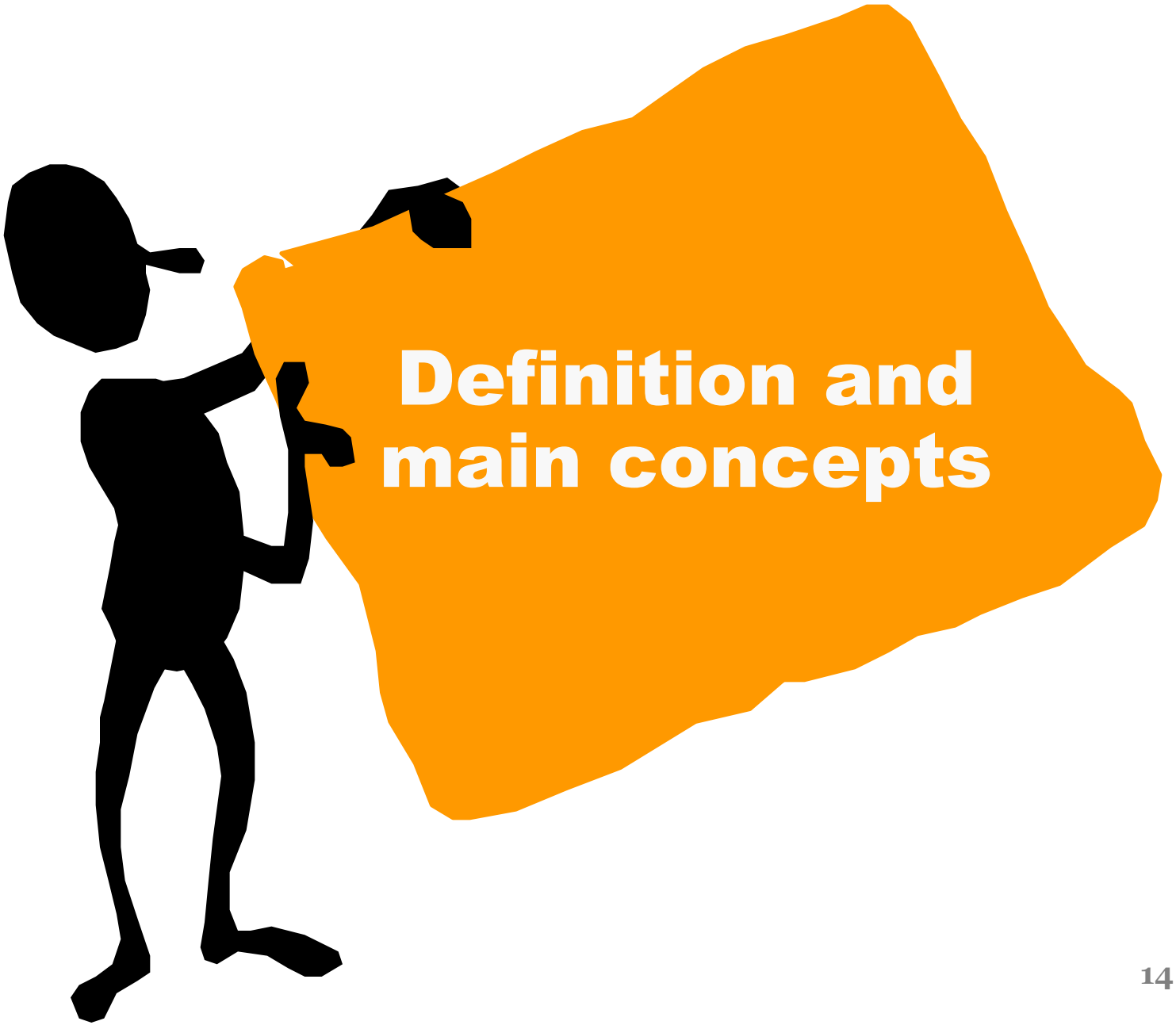
Motivation and Background



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten

Question

- What challenges did you discuss in Lecture 6 that are specific to multi-core processors?
- Are those affected by an increased number of cores?


Definition and
main concepts

What is a many-core processor?

What is a many-core processor?

"Many-core processors are defined as chips with several tens, but more likely hundreds, or even thousands of processor cores"

András Vajda

András Vajda. Programming Many-Core Chips. ISBN: 978-1-4419-9738-8 (Print) 978-1-4419-9739-5 (Online)

What is a many-core processor?

"Many-core processors are defined as chips with several tens, but more likely hundreds, or even thousands of processor cores"

András Vajda

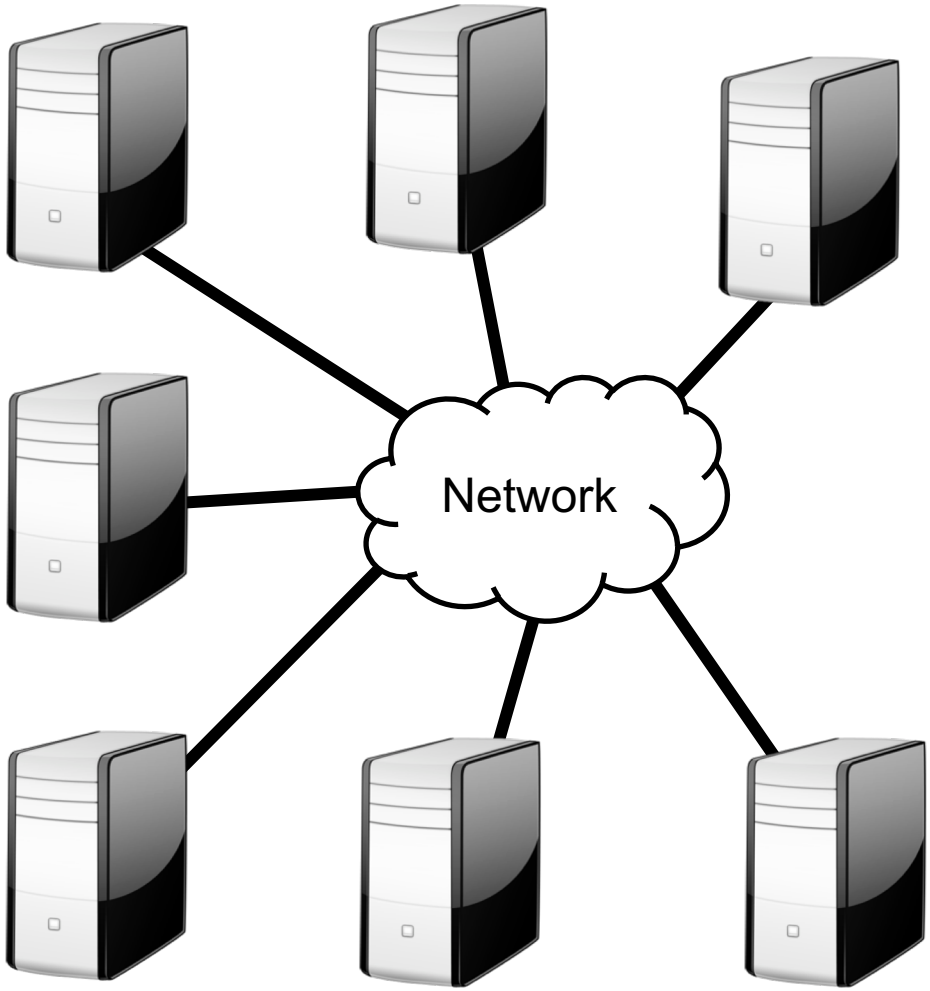
"A many-core platform, it is like an entire distributed system squeezed into a single chip."

Vincent Nelis

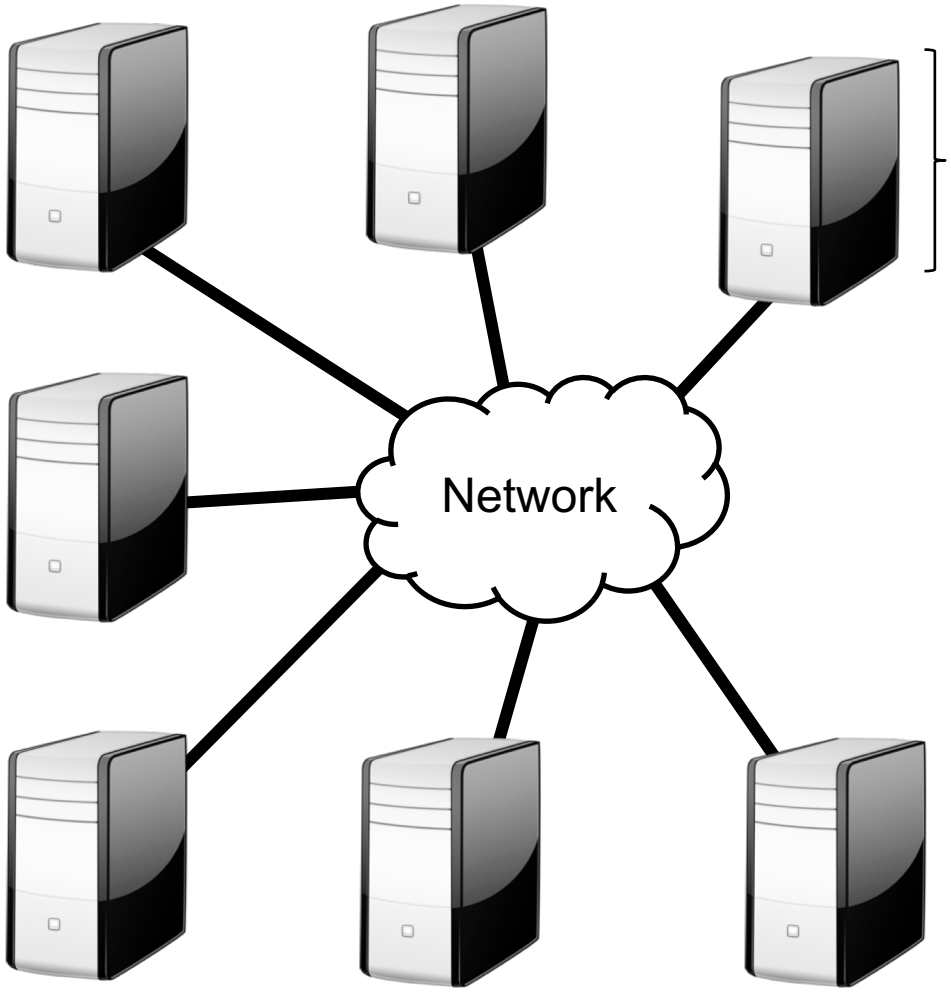
András Vajda. Programming Many-Core Chips. ISBN: 978-1-4419-9738-8 (Print) 978-1-4419-9739-5 (Online)

Vincent Nelis "The Challenge of Time-Predictability in Modern Many-Core Architectures", WCET 2014
http://www.uni-ulm.de/fileadmin/website_uni_ulm/wcet2014/slides/7.pptx

The many-core architecture

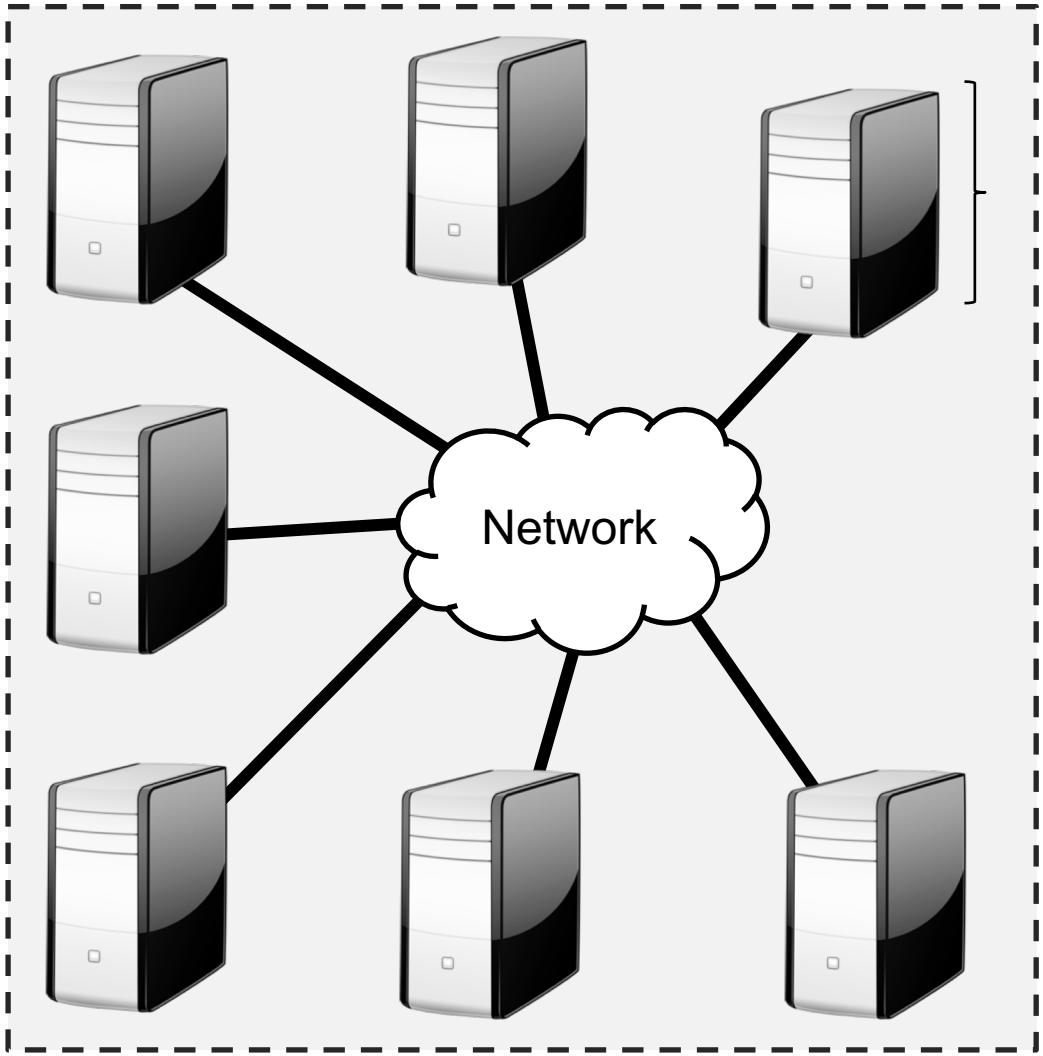


The many-core architecture



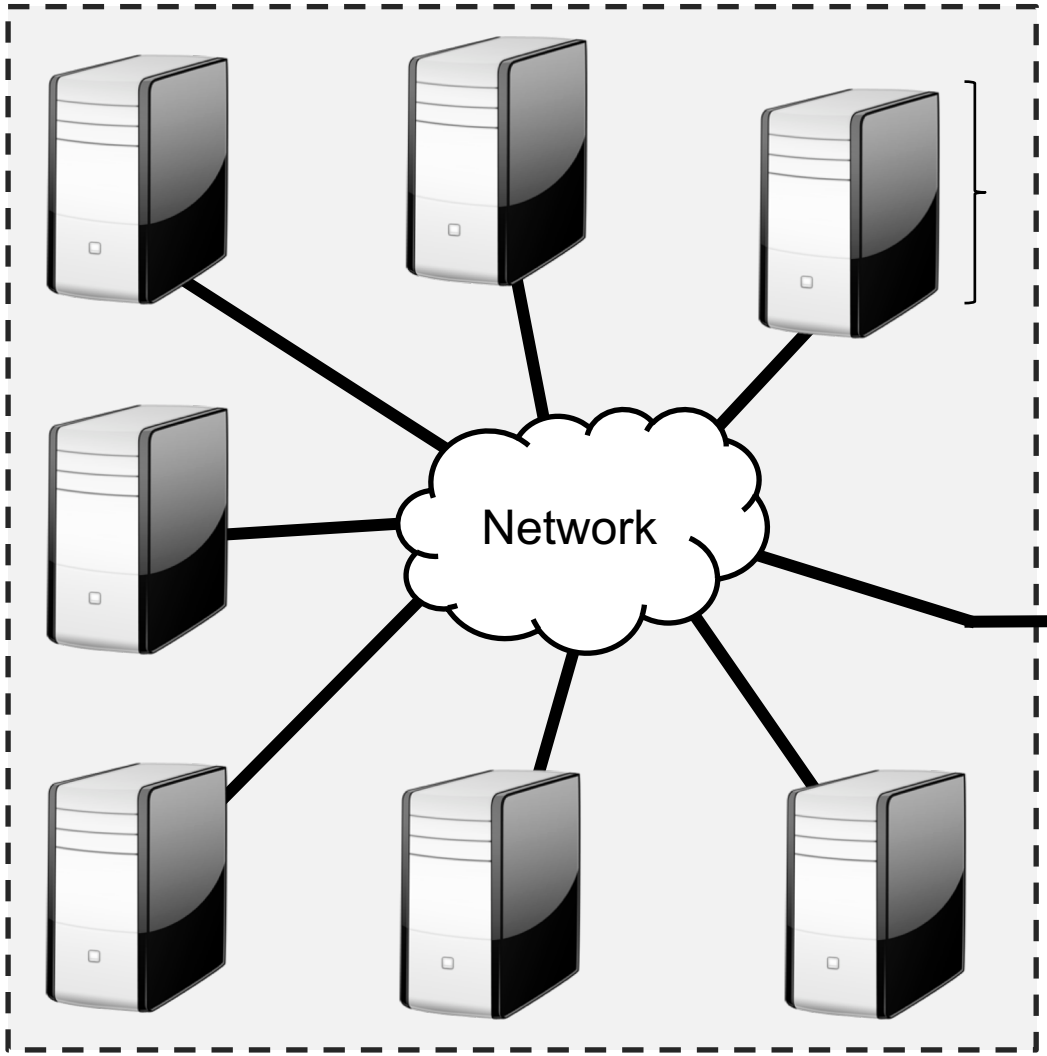
- Processing unit(s)
- Local Memory(s)
- Network Interface

The many-core architecture

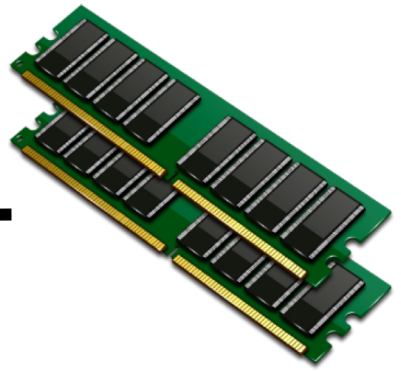


- Processing unit(s)
- Local Memory(s)
- Network Interface

The many-core architecture

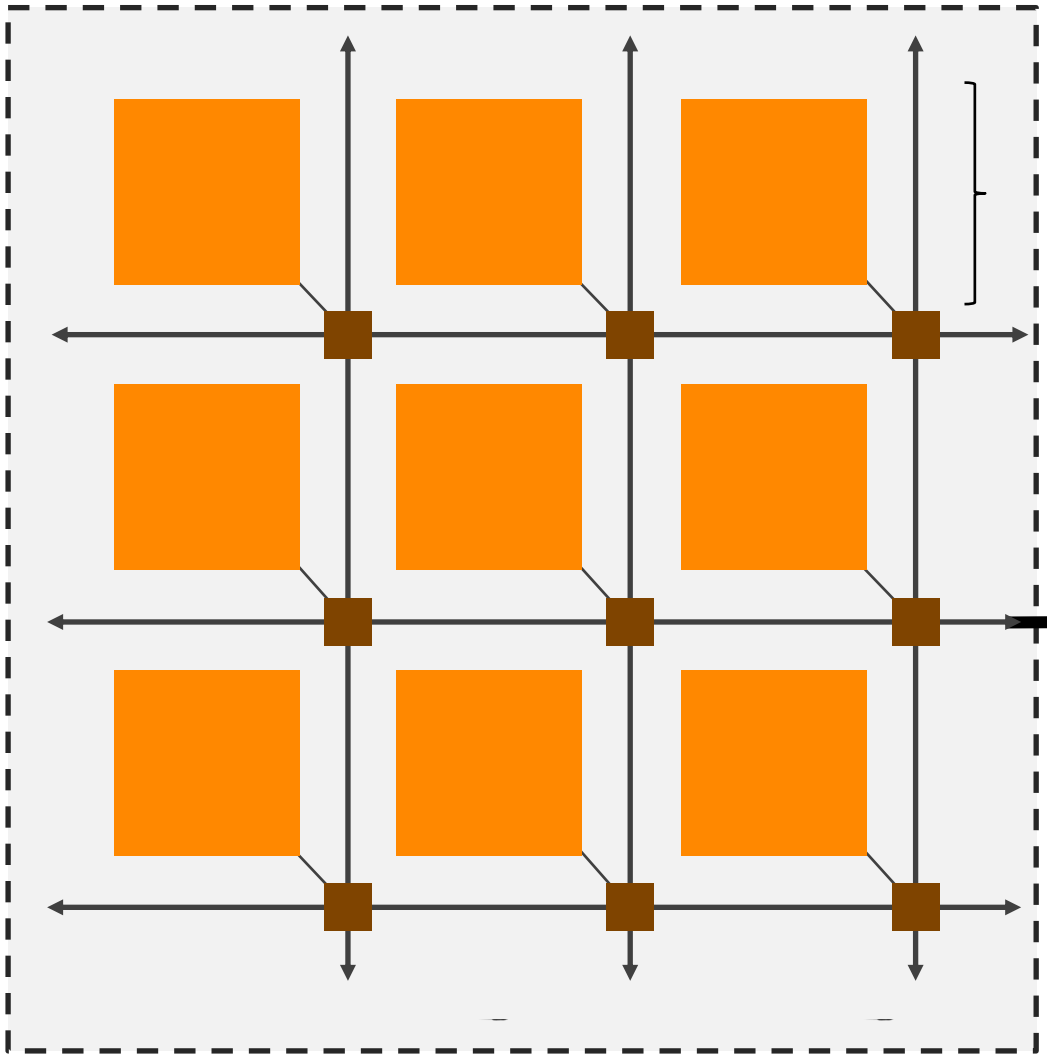


- Processing unit(s)
- Local Memory(s)
- Network Interface

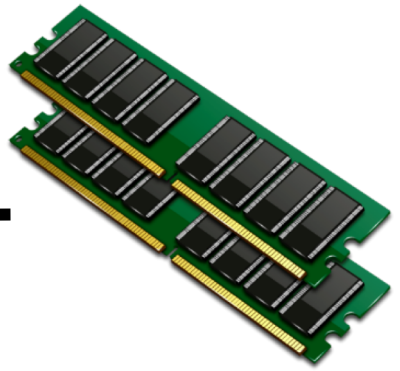


External memory

The many-core architecture

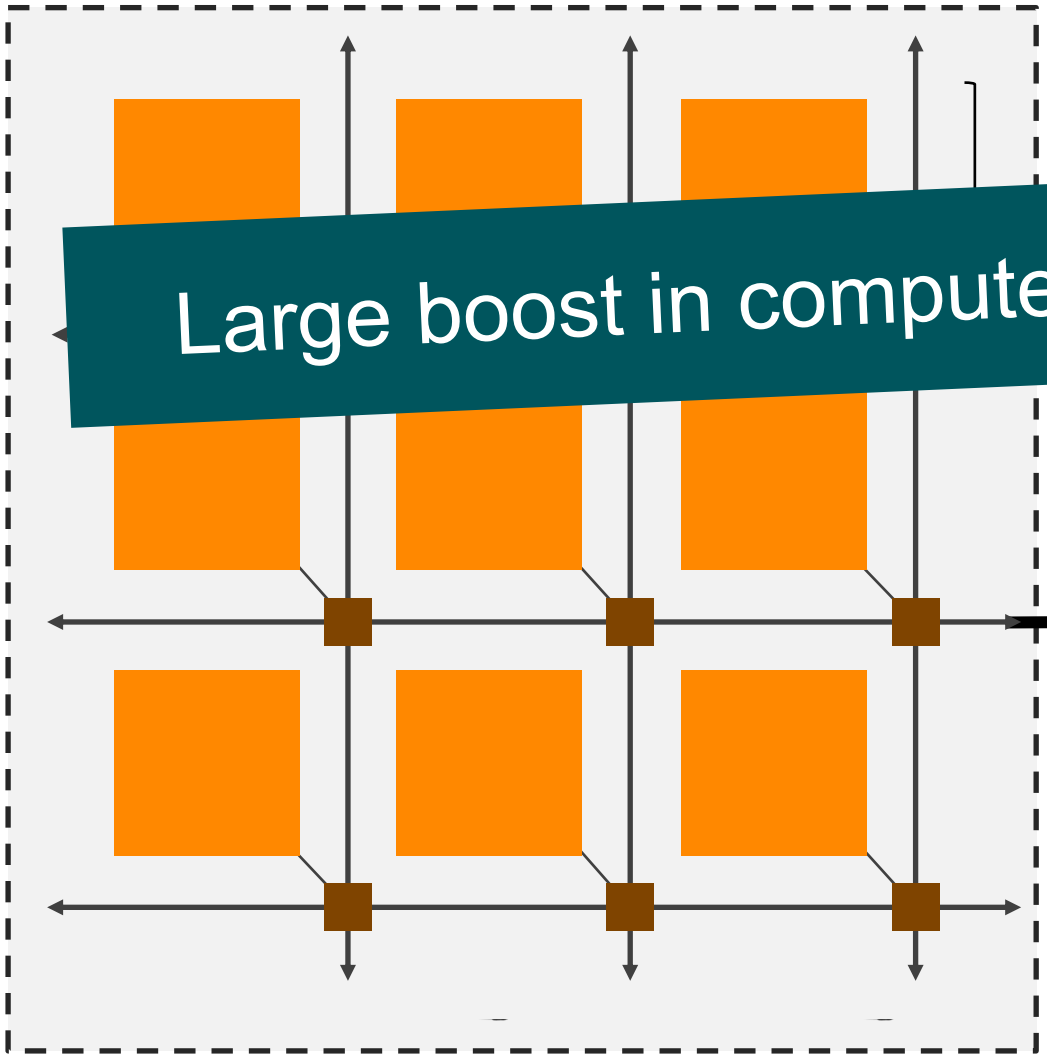


- Processing unit(s)
- Local Memory(s)
- Network Interface



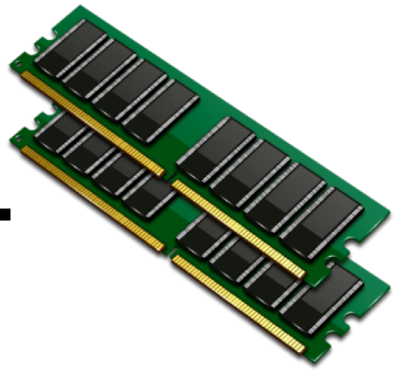
External memory

The many-core architecture



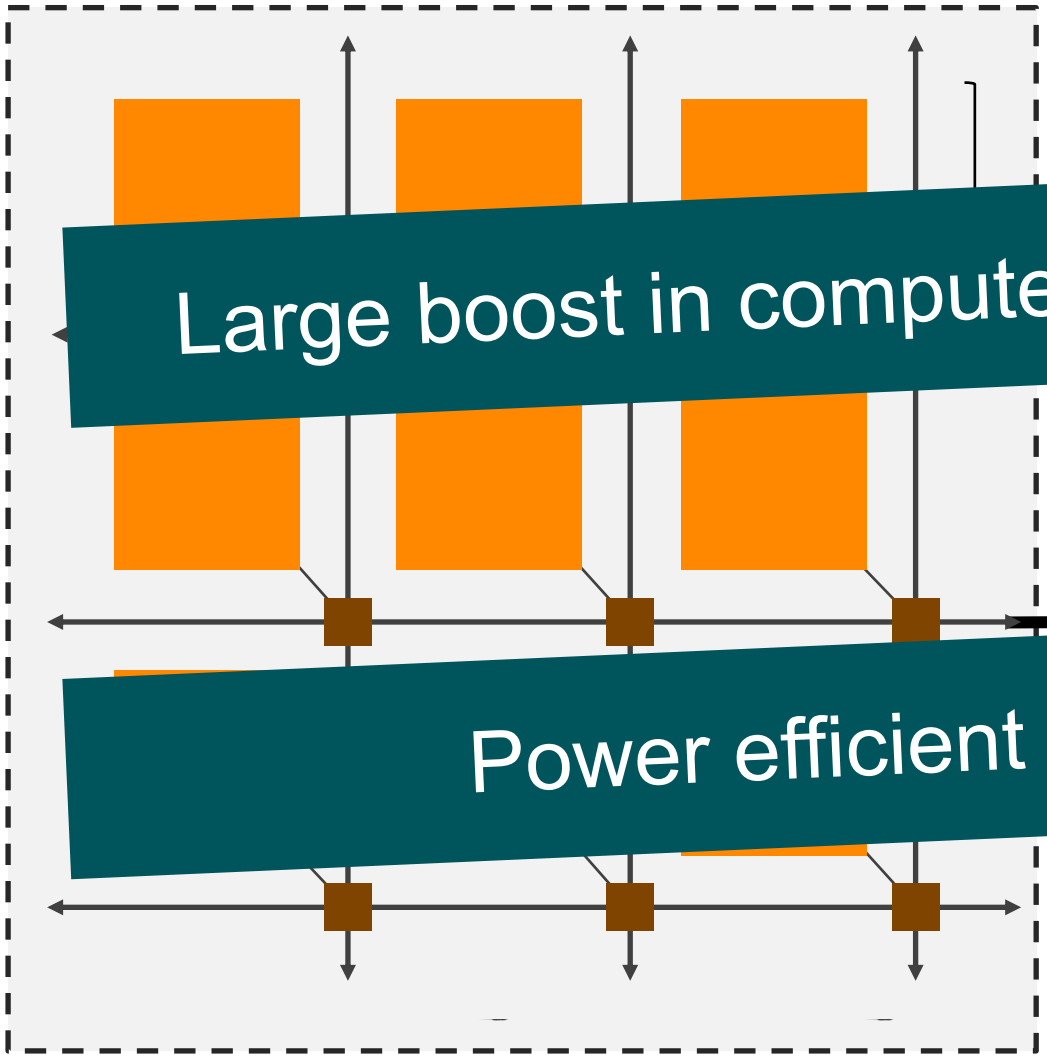
- Processing unit(s)

Large boost in compute power



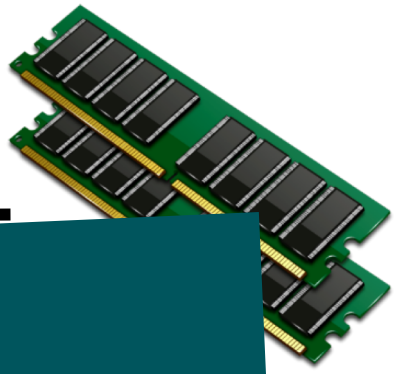
External memory

The many-core architecture



- Processing unit(s)

Large boost in compute power



Power efficient

What is a heterogeneous architecture and computing?

What is a heterogeneous architecture and computing?

- Heterogeneous computing is the well-orchestrated and coordinated effective use of a suite of diverse high-performance machines (including parallel machines) to provide super speed processing for computationally demanding tasks with diverse computing needs.

  R. Freund and D. Conwell

A. Khokhar et al., *Heterogeneous Computing: Challenges and Opportunities*

What is a heterogeneous architecture and computing?

- Heterogeneous computing is the well-orchestrated and coordinated effective use of a suite of diverse high-performance machines (including parallel machines) to provide super speed processing for computationally demanding tasks with diverse computing needs.

  R. Freund and D. Conwell

- A heterogeneous architecture is a computing system architecture in which processors use more than one instruction set, all of which share a single memory.

  Gartner dictionary

A. Khokhar et al., *Heterogeneous Computing: Challenges and Opportunities*
Gartner, *https://www.gartner.com/it-glossary/heterogeneous-architecture/*

The Heterogeneous Architecture

- **Heterogeneous Architecture** is a computing system architecture in which processors use more than one instruction set, all of which share a single memory.

The Heterogeneous Architecture

- **Heterogeneous Architecture** is a computing system architecture in which processors use more than one instruction set, all of which share a single memory.
  - Combinations of processors
    - CPU(A) + CPU(B)
    - CPU + GPU
    - CPU + FPGA
    - CPU + DSP etc.

The Heterogeneous Architecture

- **Heterogeneous Architecture** is a computing system architecture in which processors use more than one instruction set, all of which share a single memory.
  - Combinations of processors
    - CPU(A) + CPU(B)
    - C
    - C
    - C

in wide definition

Heterogeneous processor

The Heterogeneous Architecture

- **Heterogeneous Architecture** is a computing system architecture in which processors use more than one instruction set, all of which share a single memory.
  - Combinations of processors
    - CPU(A) + CPU(B)
    - CPU + GPU
    - CPU + FPGA
    - CPU + DSP etc.

- **Heterogeneous Processor**

The Heterogeneous Architecture

- **Heterogeneous Architecture** is a computing system architecture in which processors use more than one instruction set, all of which share a single memory.
  - Combinations of processors
    - CPU(A) + CPU(B)
    - CPU + GPU
    - CPU + FPGA
    - CPU + DSP etc.

- **Heterogeneous Processor**
  - A heterogeneous processor consists of different type of processing units which are located on the SoC (System on Chip), i.e., on the same die.

The Heterogeneous Architecture

- **Heterogeneous Architecture** is a computing system architecture in which processors use more than one instruction set, all of which share a single memory.
  - Combinations of processors
    - CPU(A) + CPU(B)
    - CPU + GPU
    - CPU + FPGA
    - CPU + DSP etc.

- **Heterogeneous Processor**
  - A heterogeneous processor consists of different type of processing units which are located on the SoC (System on Chip), i.e., on the same die.
  - Heterogeneous processors / Processing Units (PUs):
    - APU (Accelerated Processing Unit), CPU + iGPU (integrated GPU),
    - CPU + FPGA, CPU + iGPU + FPGA, CPU + DSP etc.

The Heterogeneous Architecture

- **Heterogeneous Architecture** is a computing system architecture in which processors use more than one instruction set, all of which share a single memory.
  - Combinations of processors
    - CPU(A) + CPU(B)
    - CPU + GPU
    - CPU + FPGA
    - CPU + DSP etc.

- **Heterogeneous Processor**
  - A heterogeneous processor consists of different type of processing units which are located on the SoC (System on Chip), i.e., on the same die.
  - Heterogeneous processors / Processing Units (PUs):
    - APU (Accelerated Processing Unit), CPU + iGPU (integrated GPU),
    - CPU + FPGA, CPU + iGPU + FPGA, CPU + DSP etc.
  - Challenges to handle memory

The Heterogeneous Processor

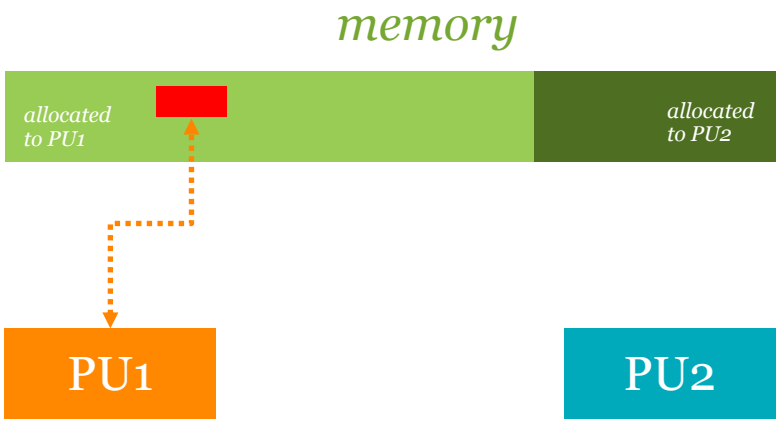
- Main concepts / Challenges

The Heterogeneous Processor

- Main concepts / Challenges
  - How to handle data between the processing units

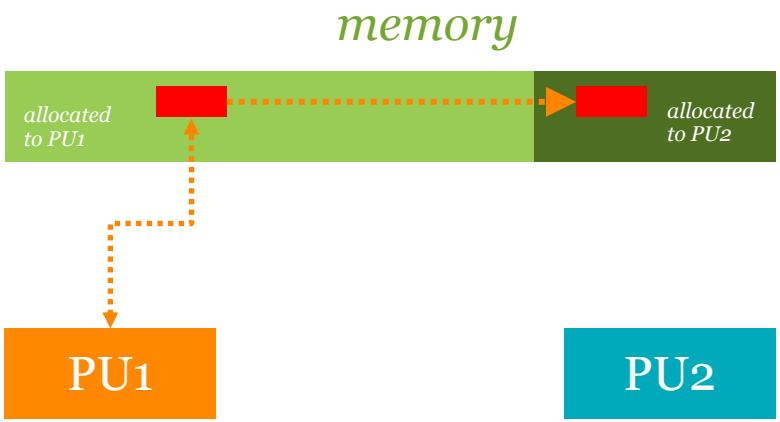
The Heterogeneous Processor

- Main concepts / Challenges
  - How to handle data between the processing units



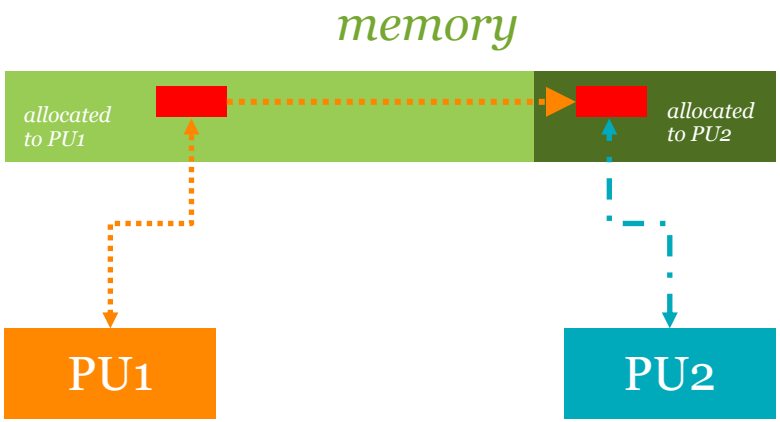
The Heterogeneous Processor

- Main concepts / Challenges
  - How to handle data between the processing units



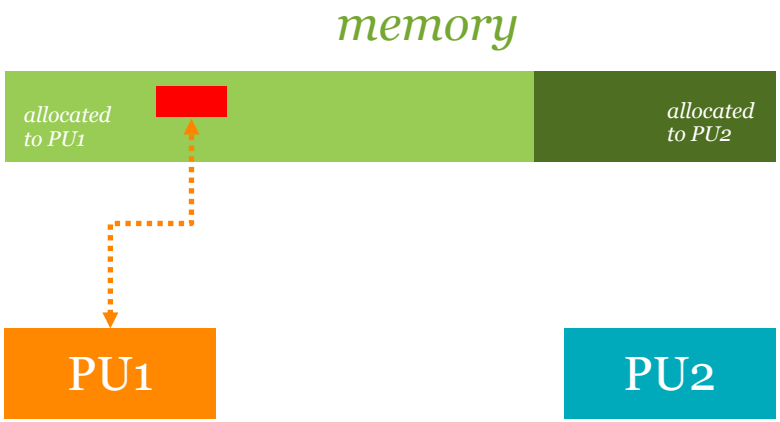
The Heterogeneous Processor

- Main concepts / Challenges
  - How to handle data between the processing units



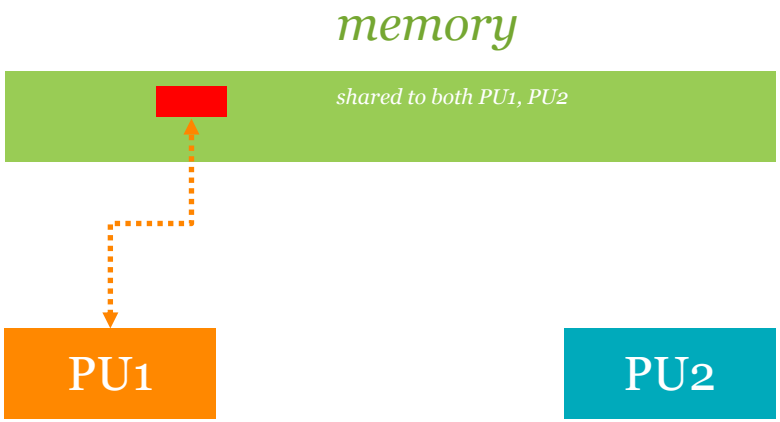
The Heterogeneous Processor

- Main concepts / Challenges
  - How to handle data between the processing units



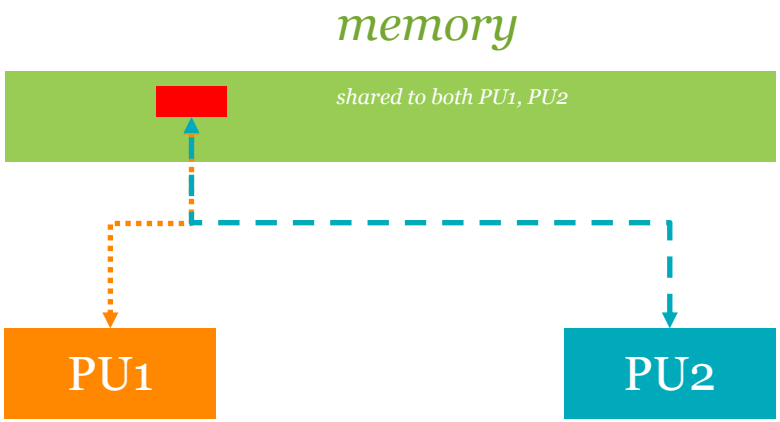
The Heterogeneous Processor

- Main concepts / Challenges
  - How to handle data between the processing units



The Heterogeneous Processor

- Main concepts / Challenges
  - How to handle data between the processing units



The Heterogeneous Processor

- Main concepts / Challenges
  - How to handle data between the processing units

The Heterogeneous Processor

- Main concepts / Challenges
  - How to handle data between the processing units

  - Parallel programming / Programming model

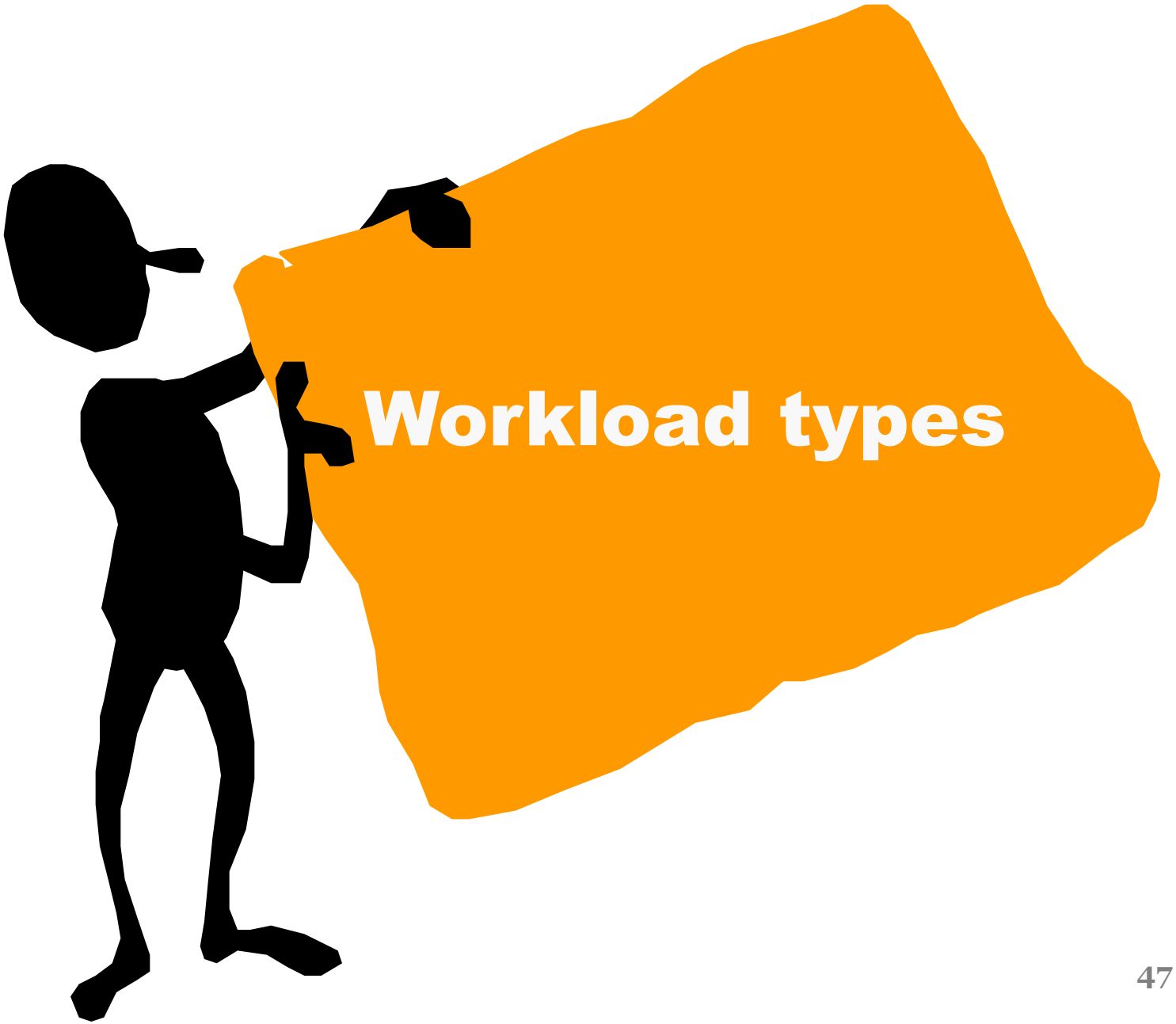
The Heterogeneous Processor

- Main concepts / Challenges
  - How to handle data between the processing units

  - Parallel programming / Programming model
  - Use cases

Question

- What types of applications would fit many-core or heterogeneous platforms?
- What differences can exist between way the computation is done?

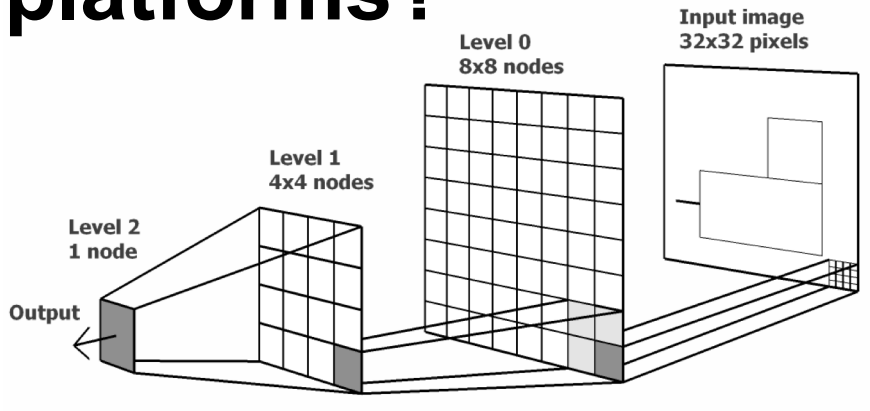

Workload types

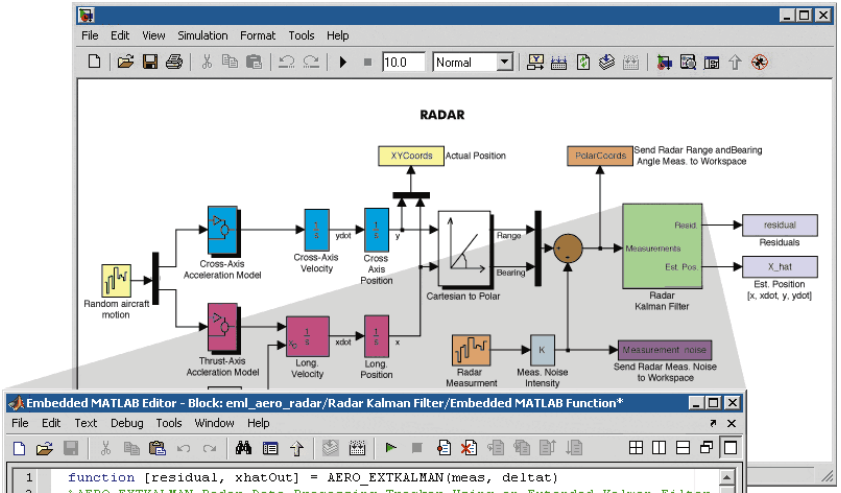
What workload types are executed on these platforms?

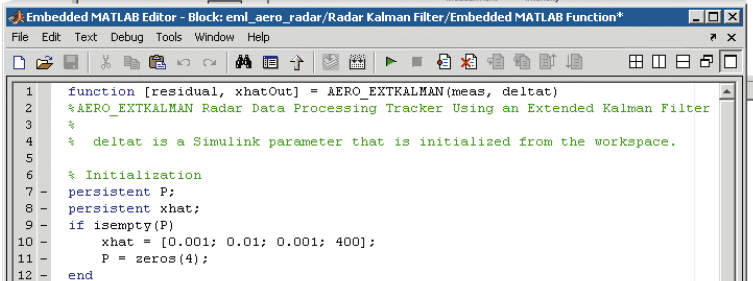
- Application types:

What workload types are executed on these platforms?

- Application types:
  - Image processing
  - Control applications
  - Infotainment
  - …

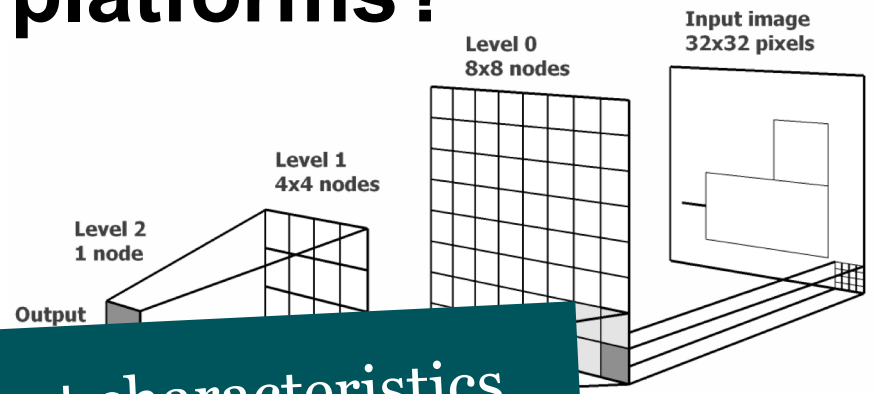




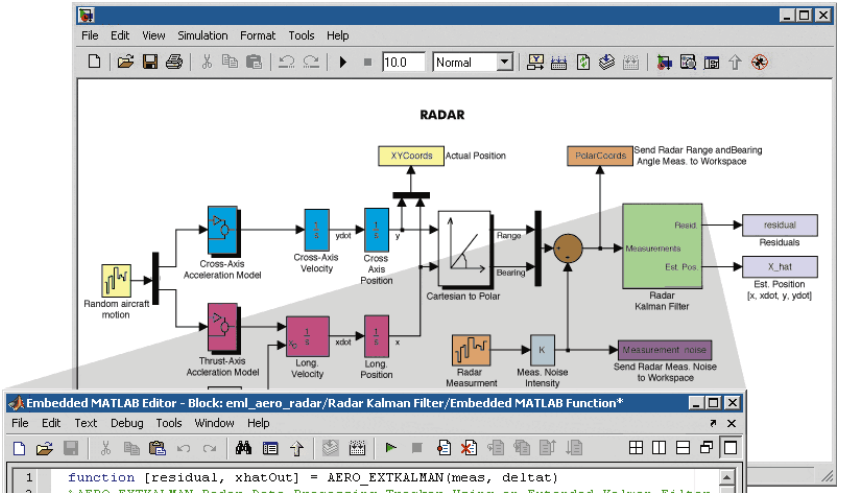


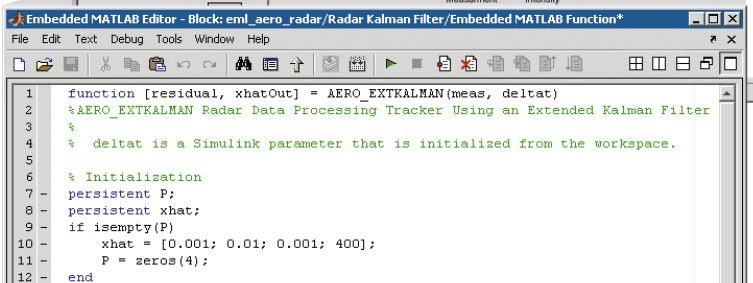
What workload types are executed on these platforms?

- Application types:
  - Image processing
  - Control applications
  - …



Workloads with different characteristics



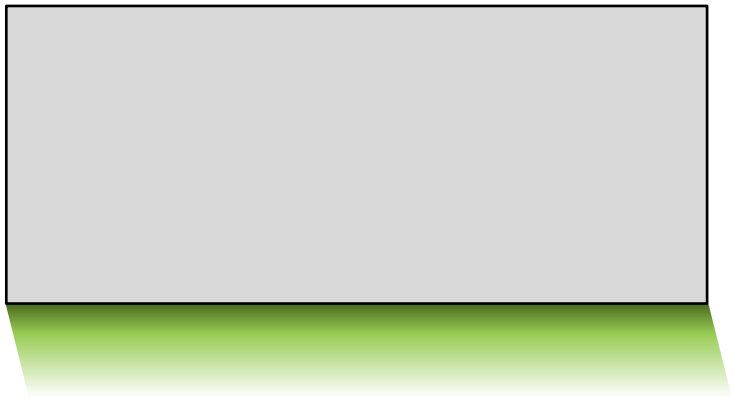


Fundamental differences in parallelism

- Parallelise work tasks
- Parallelise data
- Illustrative example
  - 2 workers

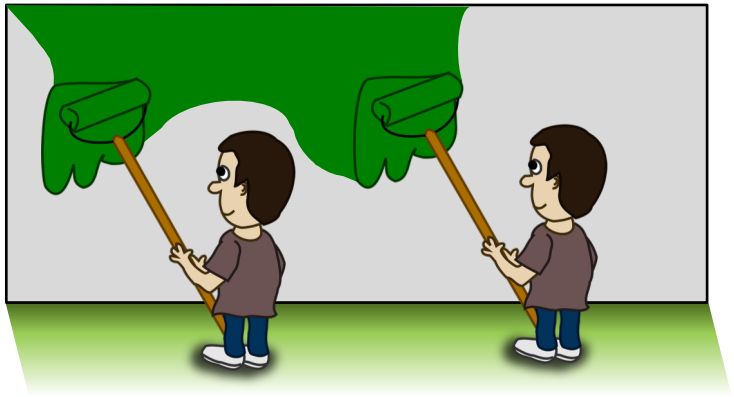
Fundamental differences in parallelism

- Parallelise work tasks
- Parallelise data
- Illustrative example
  - 2 workers



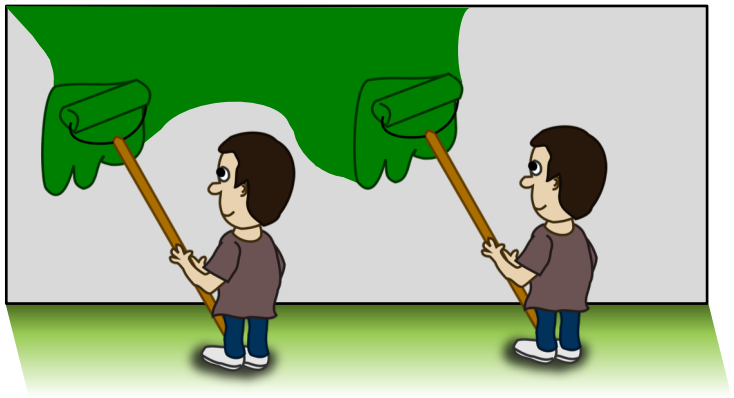
Fundamental differences in parallelism

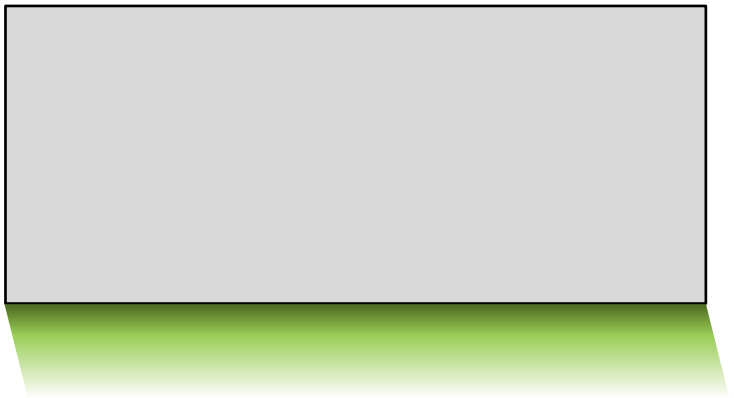
- Parallelise work tasks
- Parallelise data
- Illustrative example
  - 2 workers



Fundamental differences in parallelism

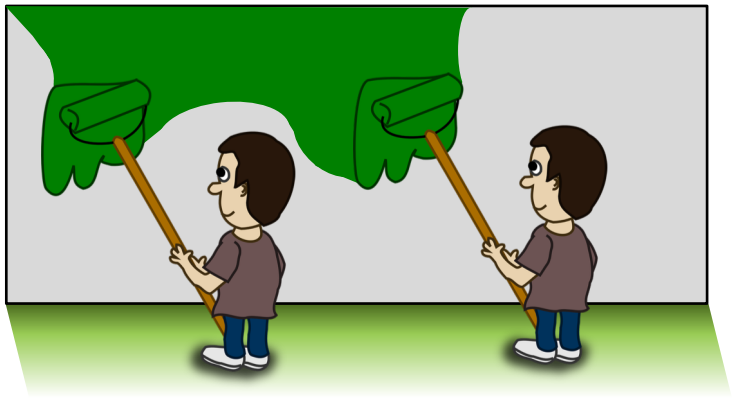
- Parallelise work tasks
- Parallelise data
- Illustrative example
  - 2 workers

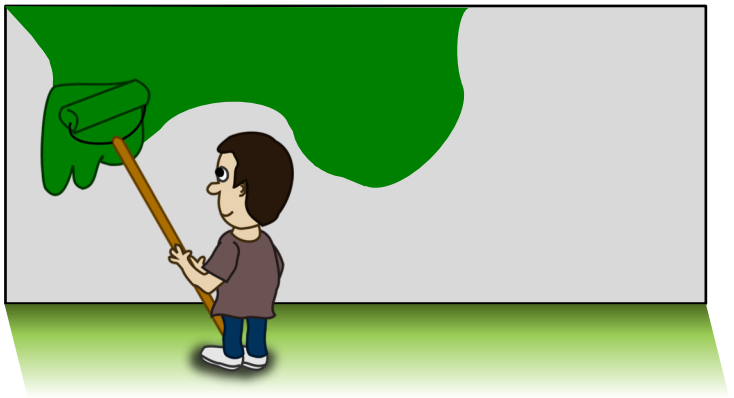




Fundamental differences in parallelism

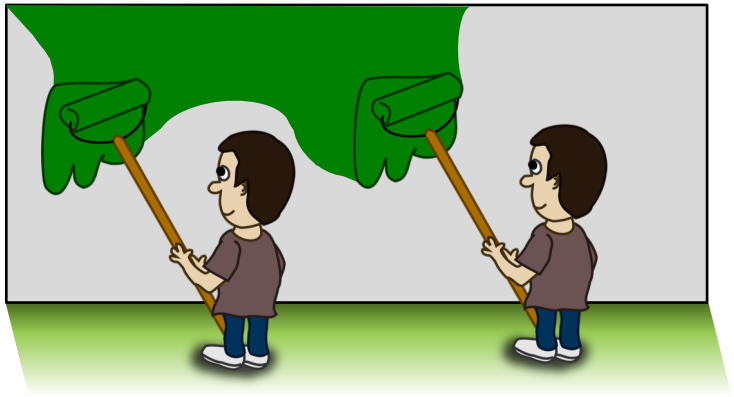
- Parallelise work tasks
- Parallelise data
- Illustrative example
  - 2 workers

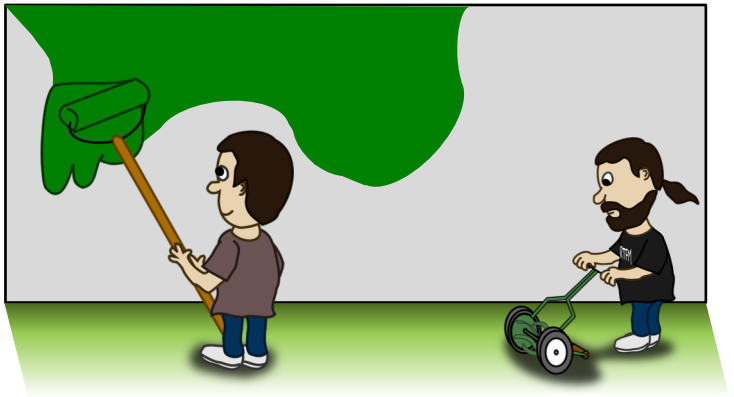




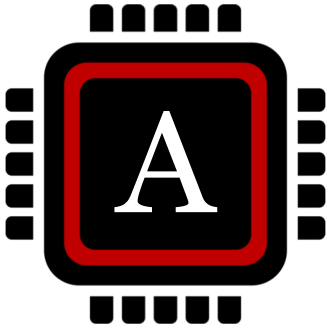
Fundamental differences in parallelism

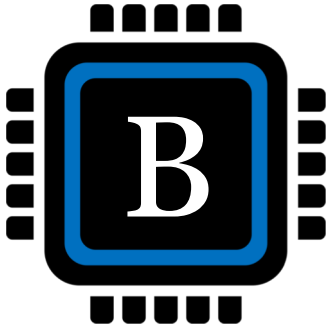
- Parallelise work tasks
- Parallelise data
- Illustrative example
  - 2 workers



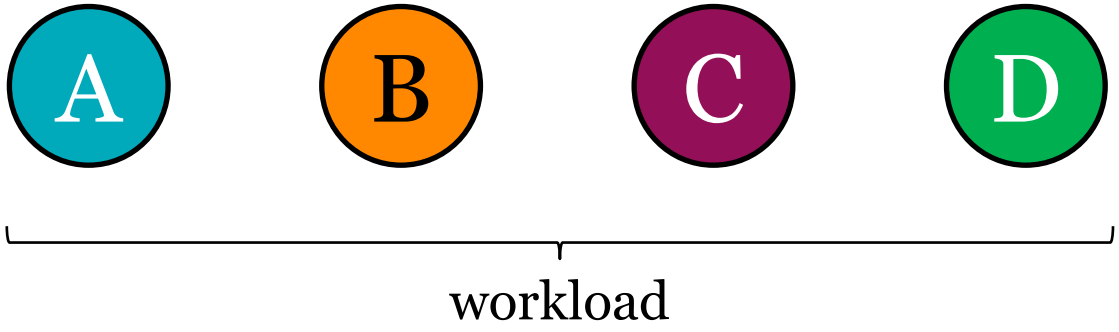


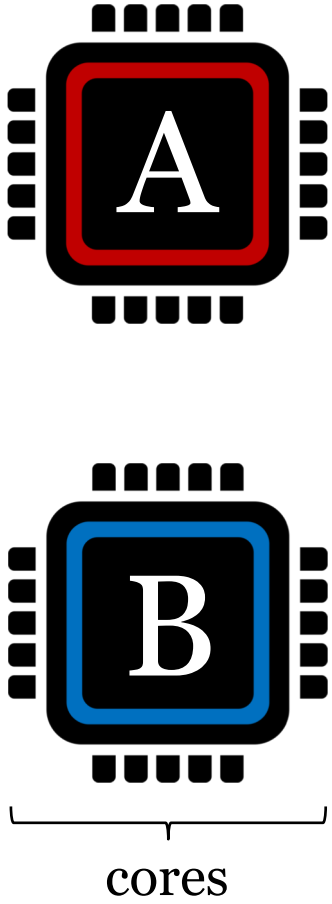
Independent Tasks



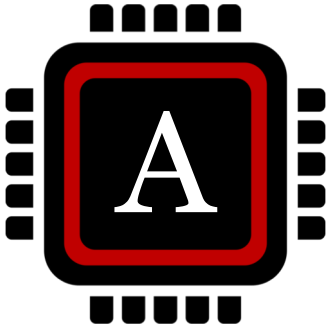


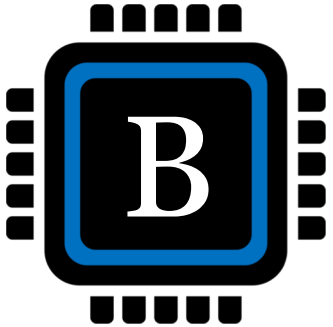
Independent Tasks I



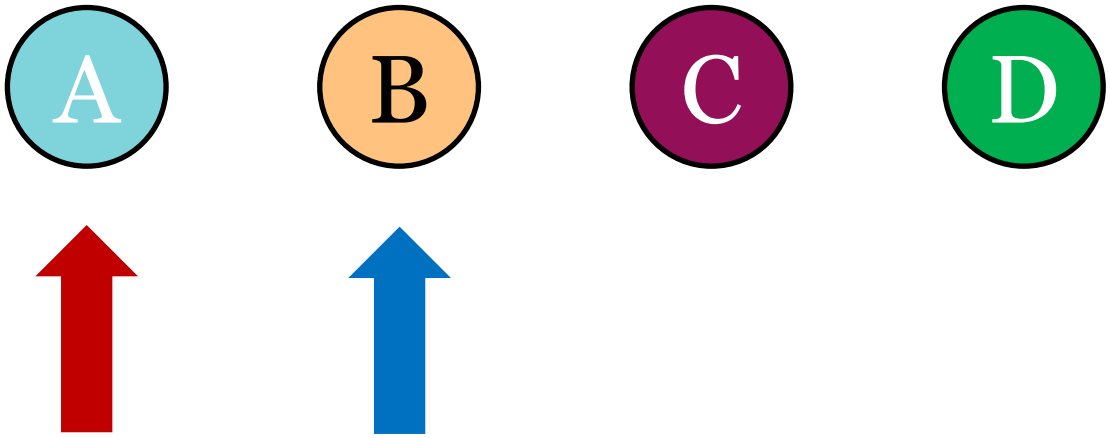


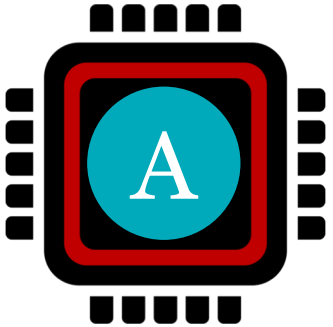
Independent Tasks

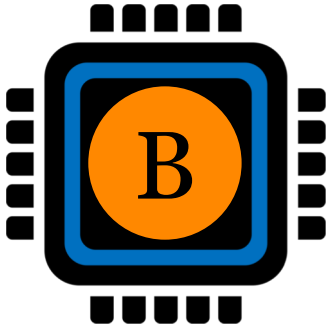




Independent Tasks I

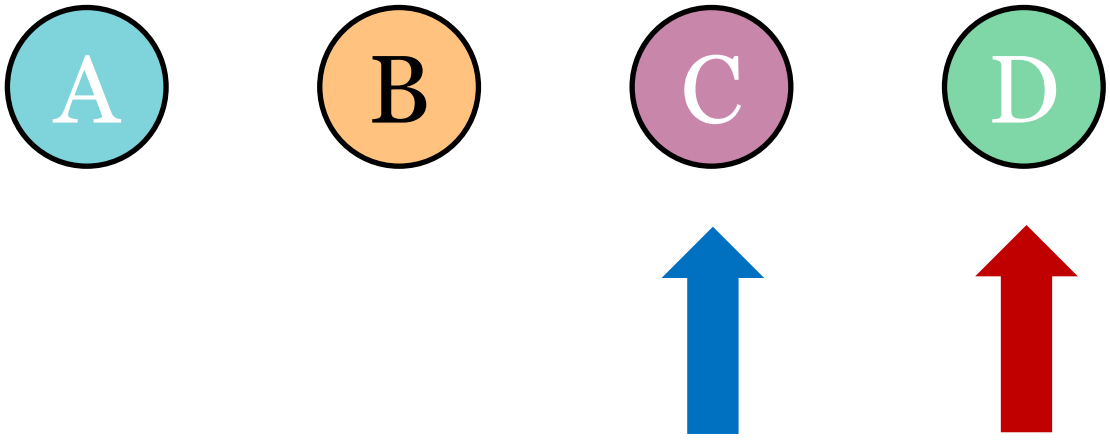


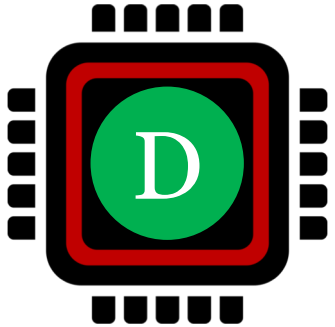


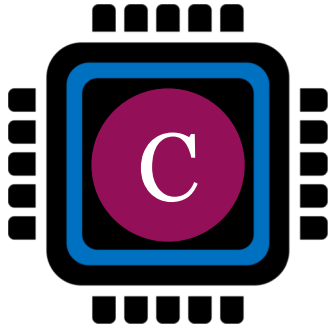


Time = 1

Independent Tasks I

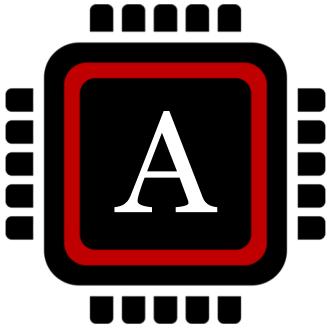


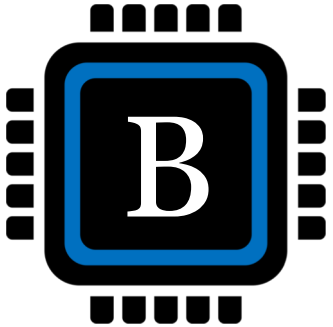




Time = 2

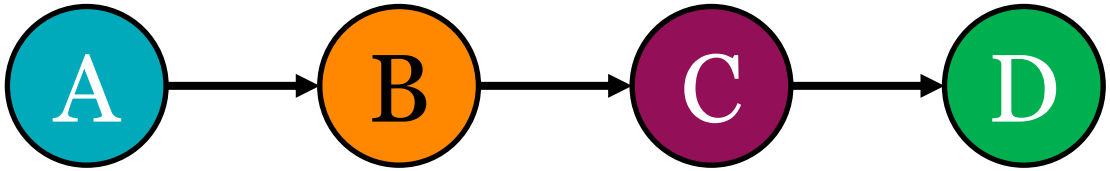
Independent Tasks I

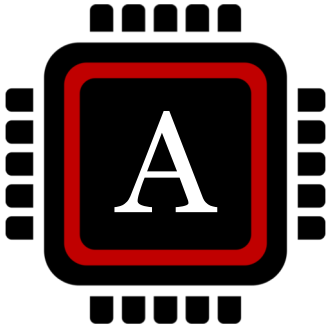


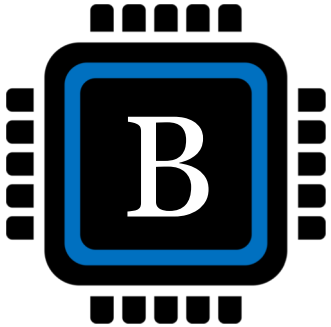


Time = 3

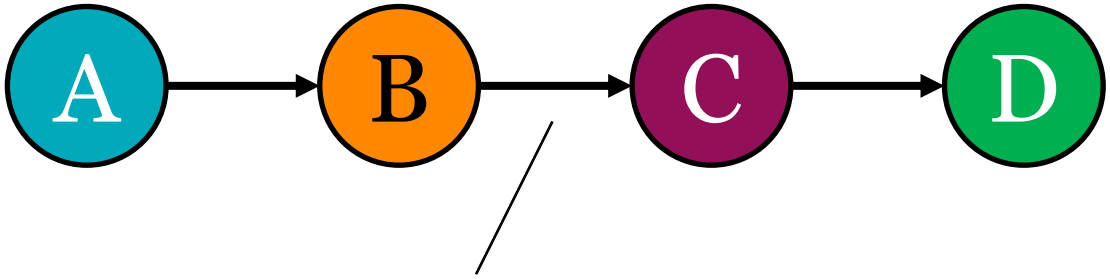
Sequential Tasks II

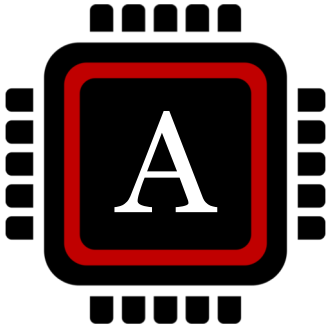


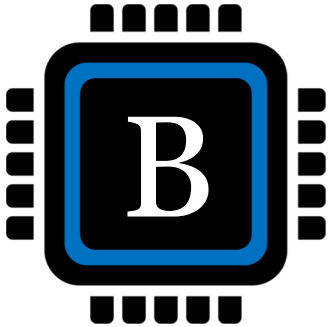




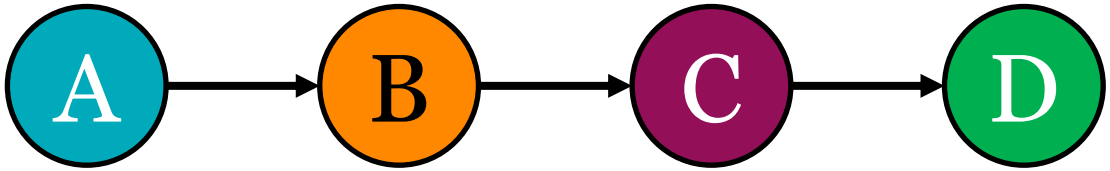
Sequential Tasks II

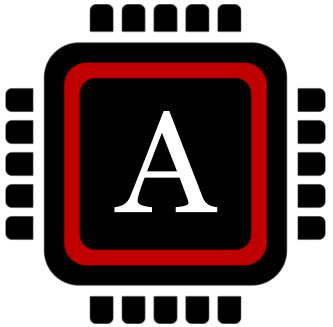


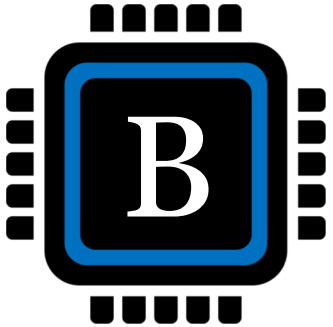




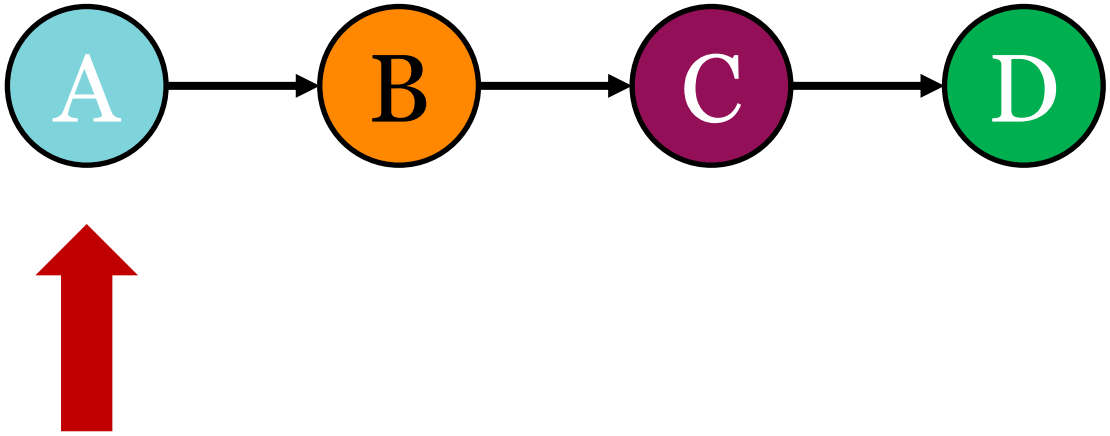
Sequential Tasks II

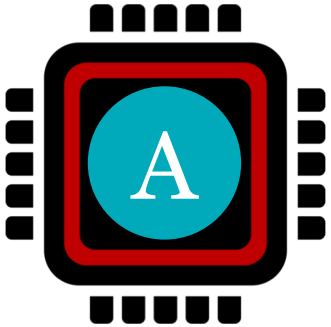


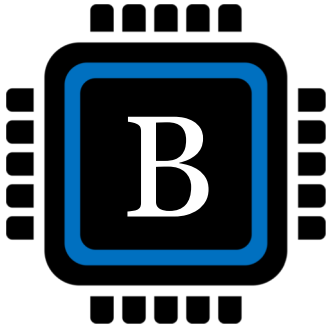




Sequential Tasks II

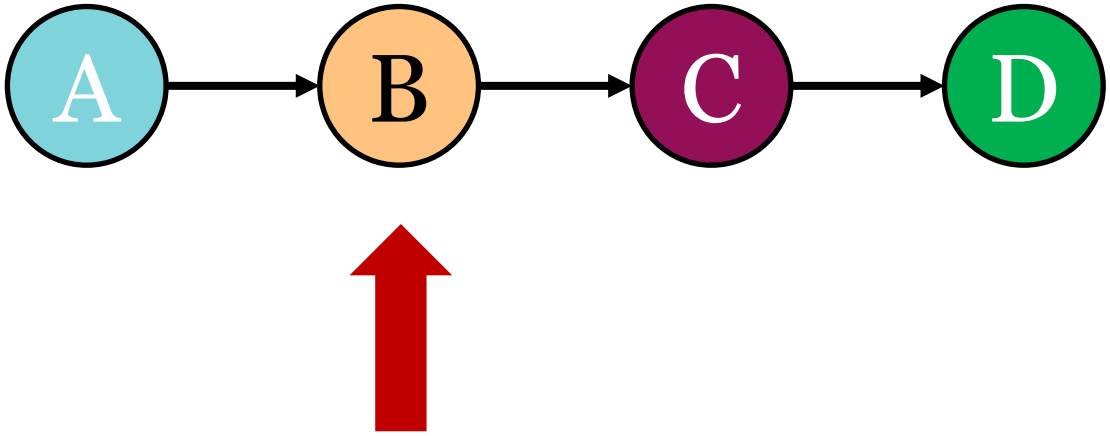


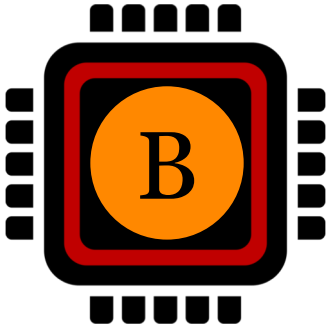


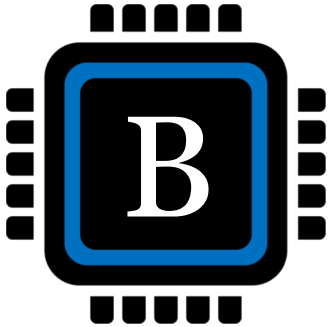


Time = 1

Sequential Tasks II

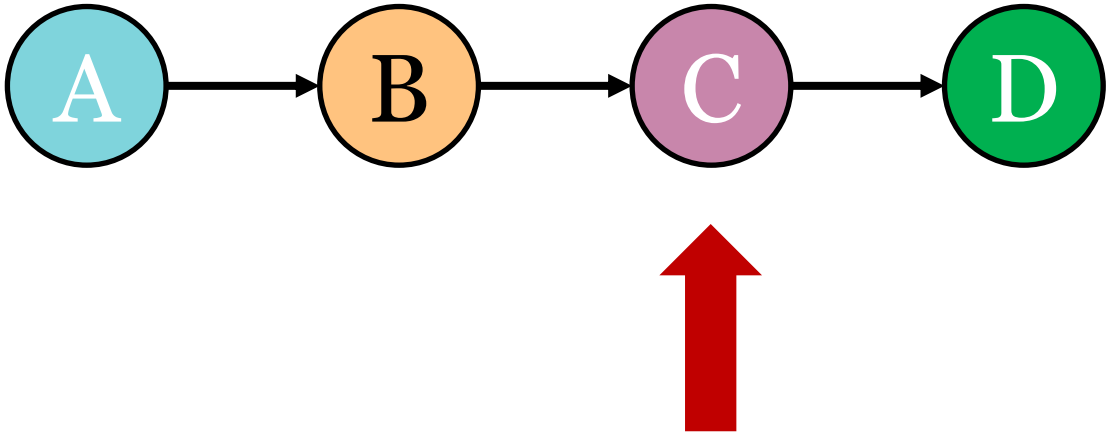


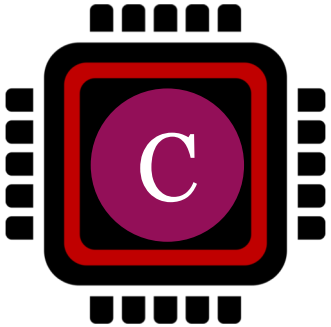


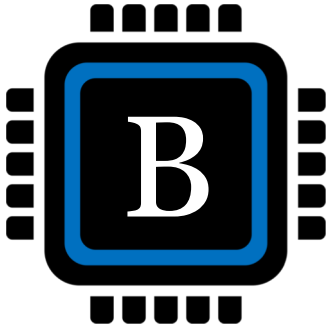


Time = 2

Sequential Tasks II

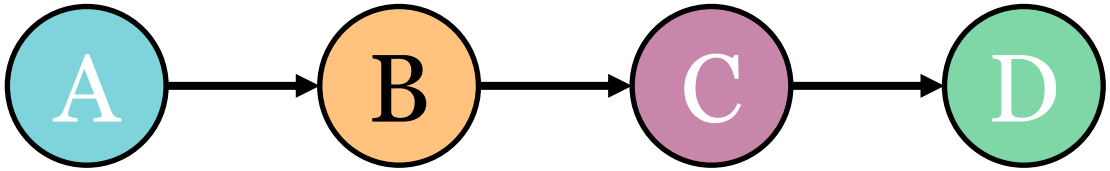


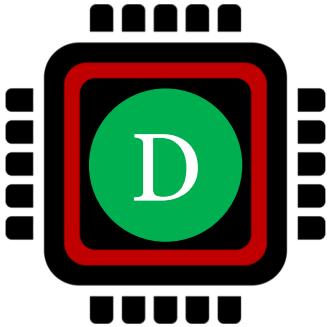


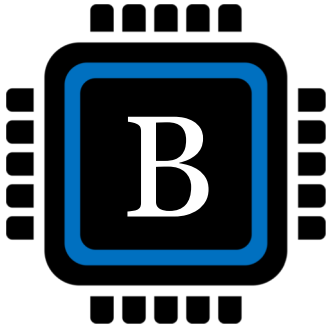


Time = 3

Sequential Tasks II

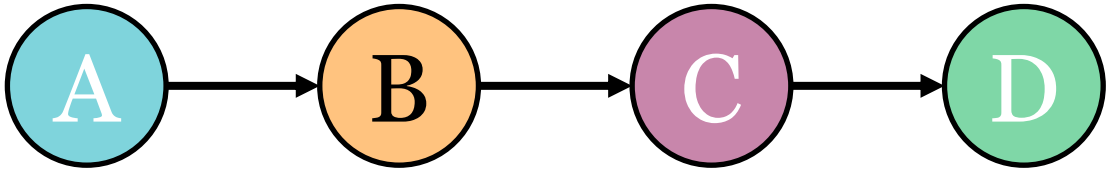


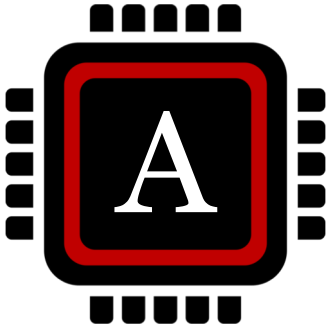


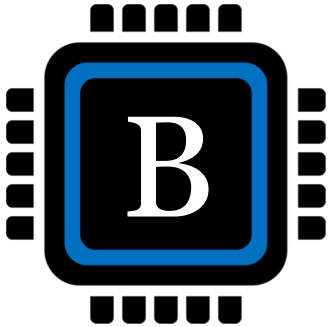


Time = 4

Sequential Tasks II



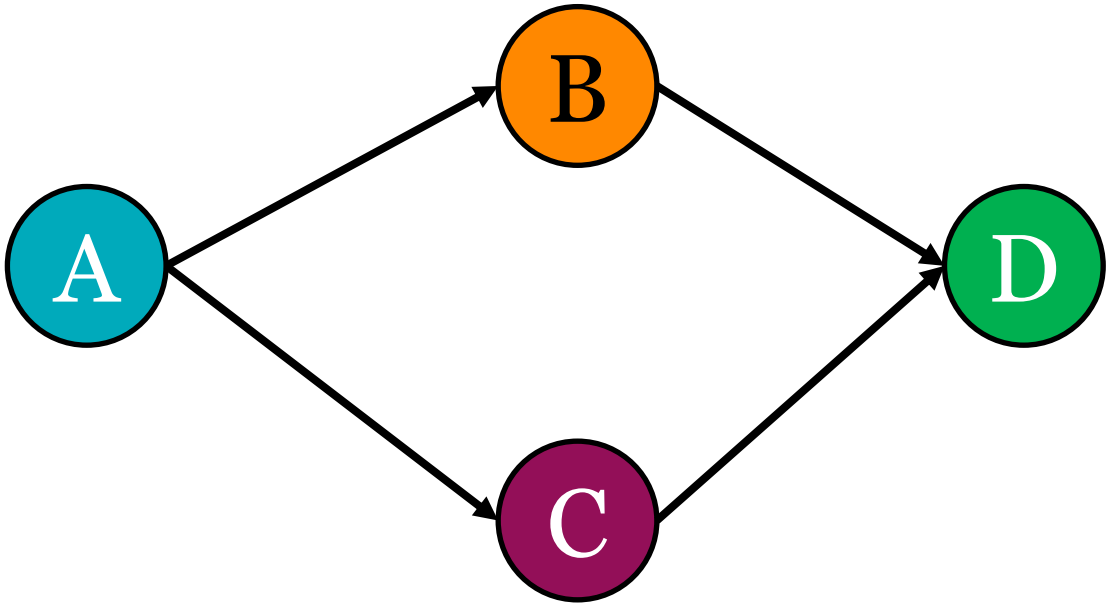


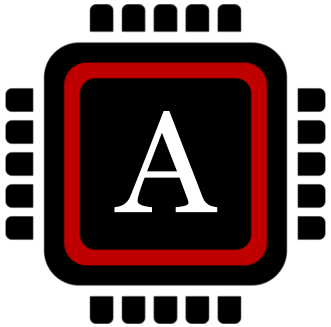


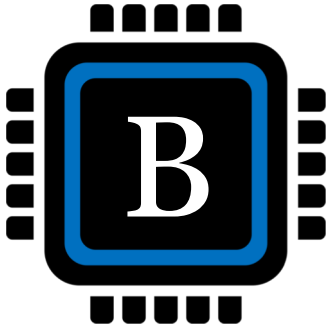
Time = 5

Parallel Tasks

III

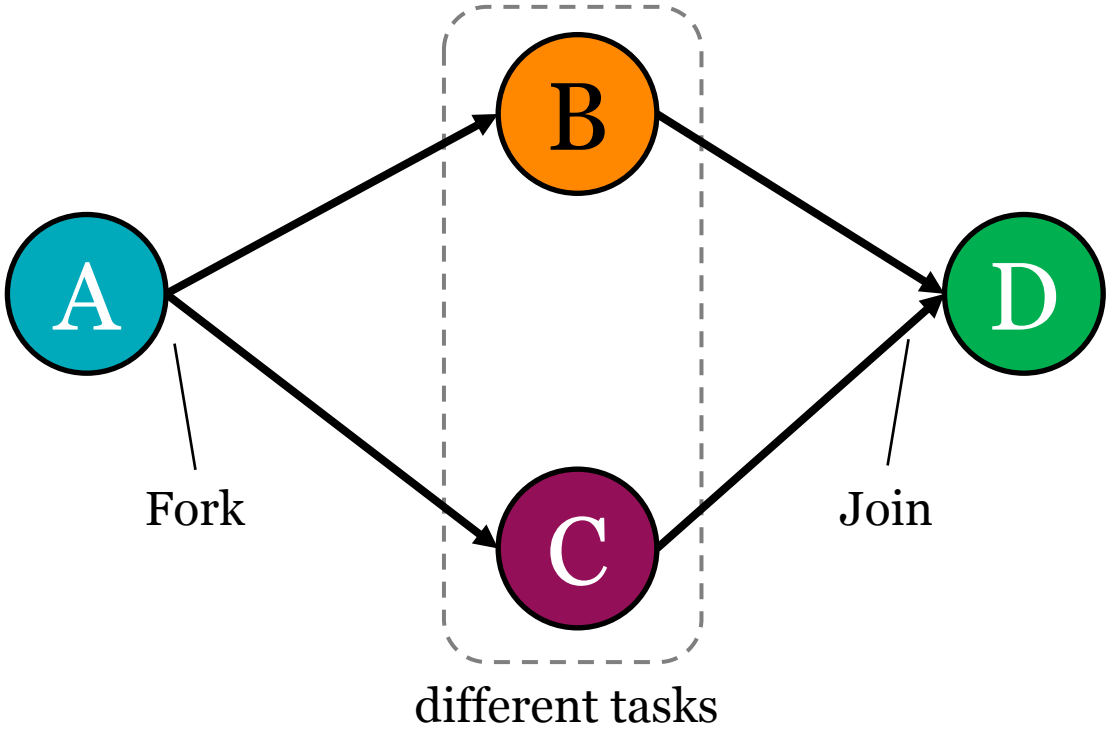


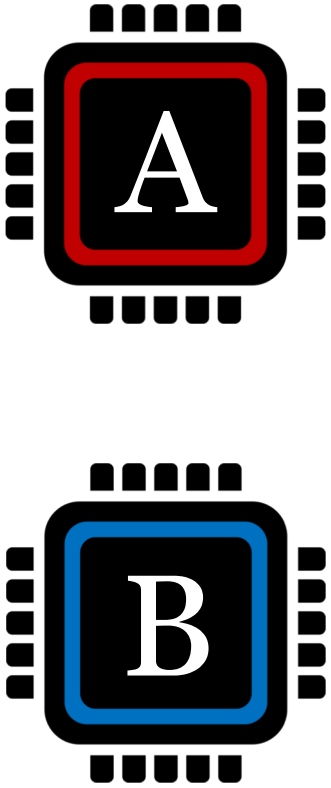




Parallel Tasks

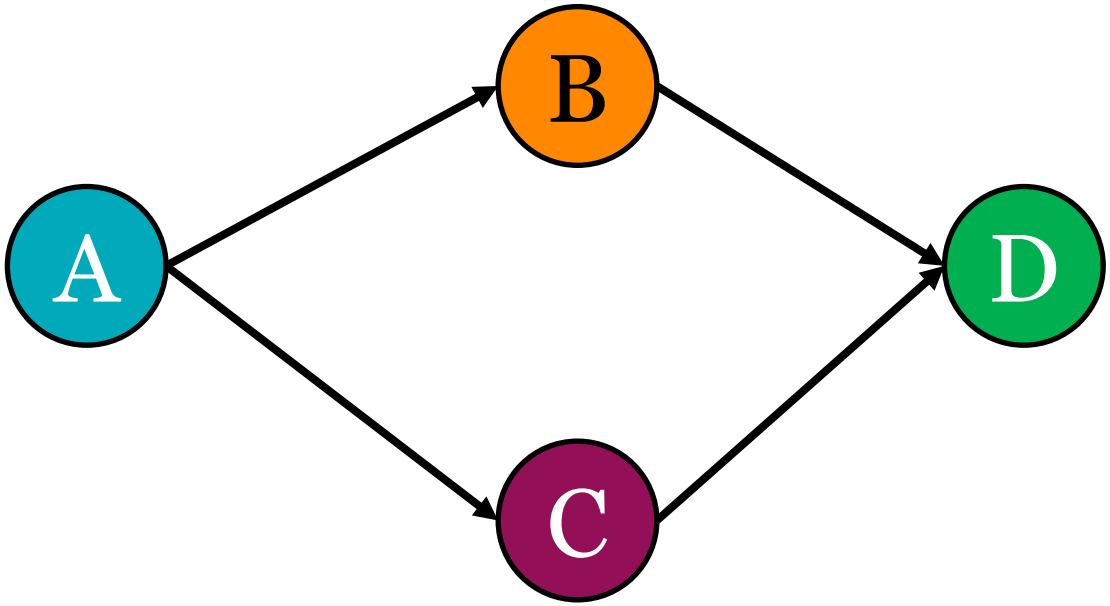
III

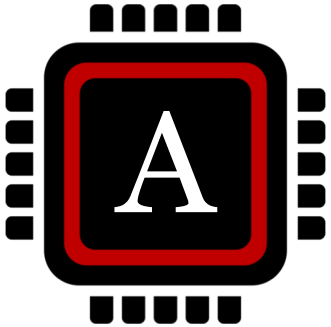


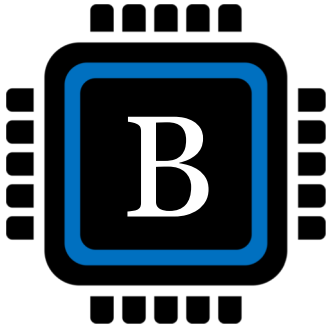


Parallel Tasks

III

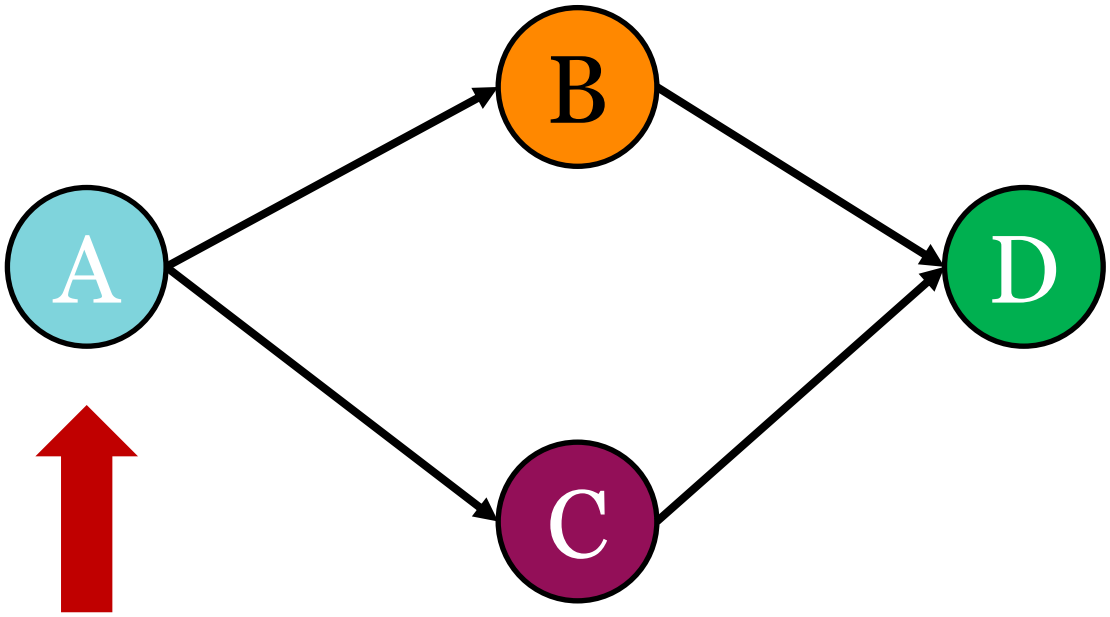


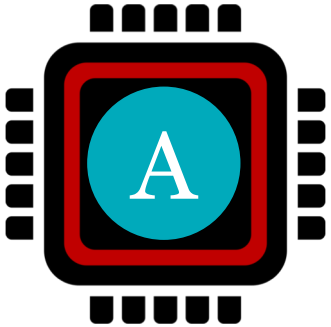


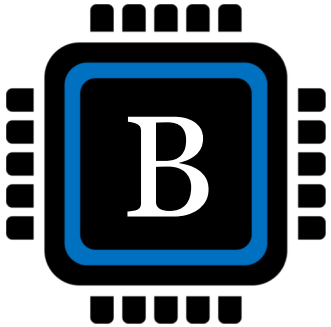


Parallel Tasks

III



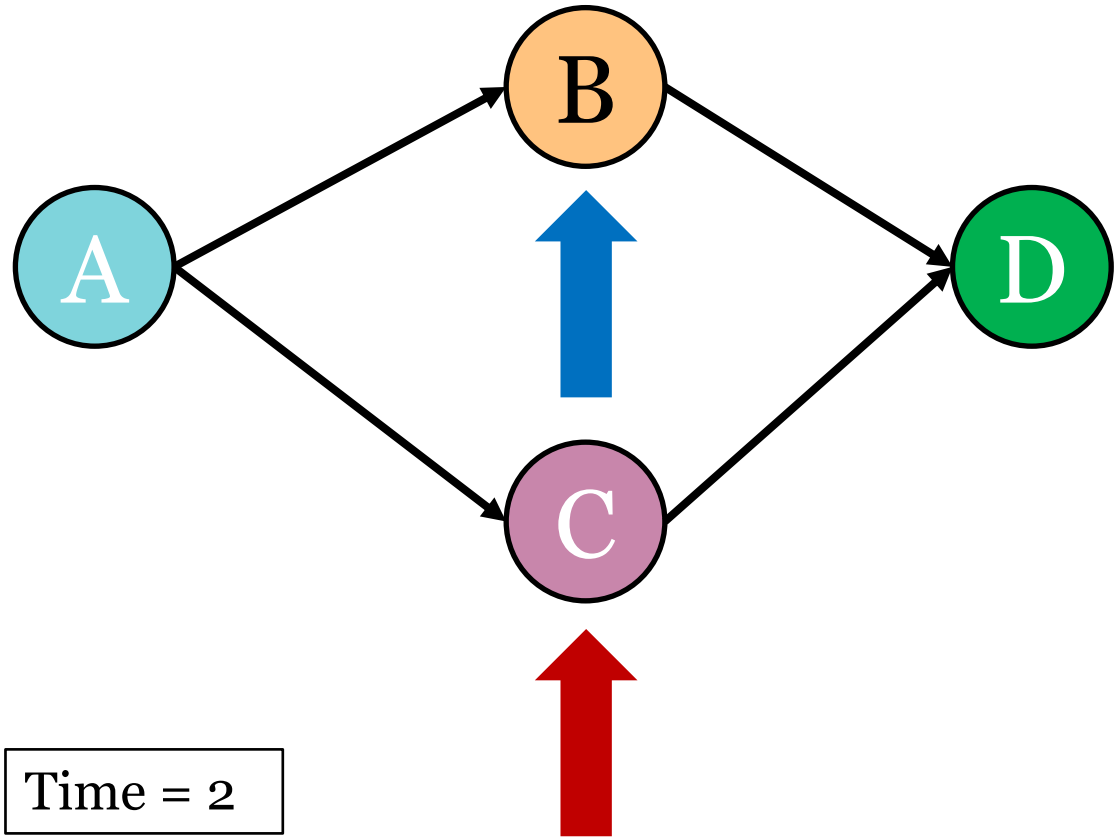


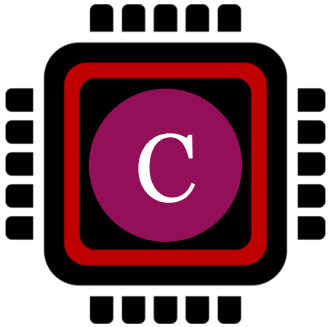


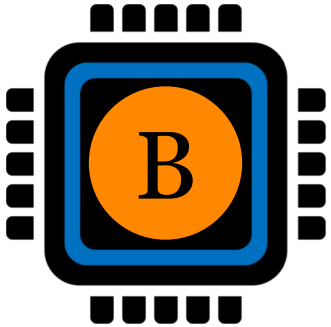
Time = 1

Parallel Tasks

III

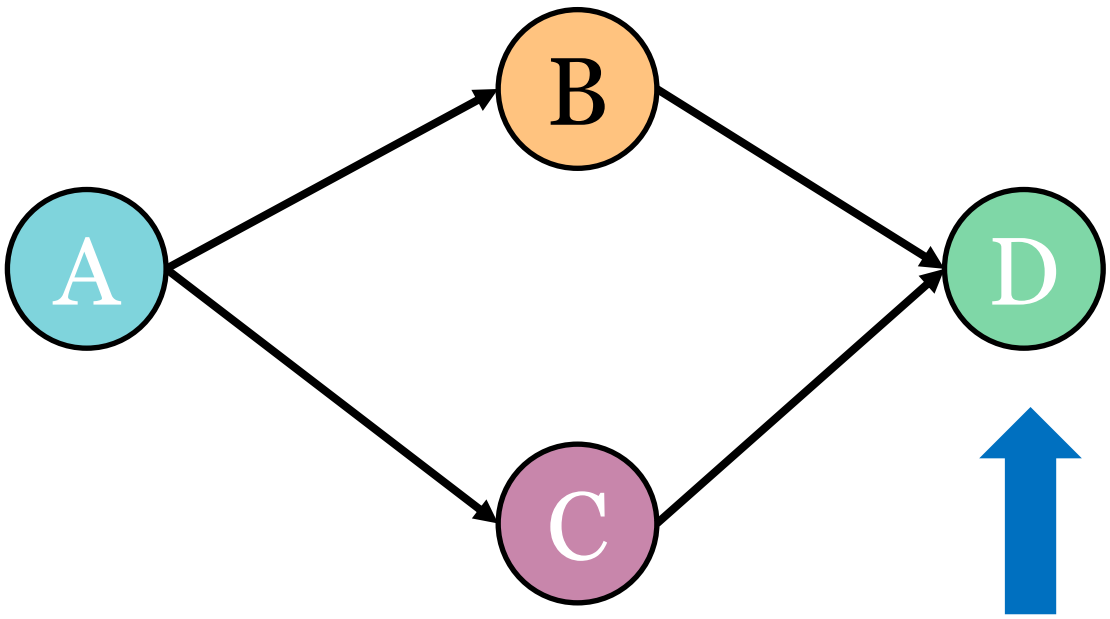


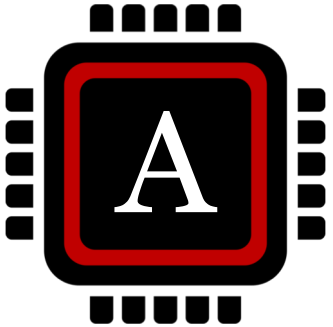


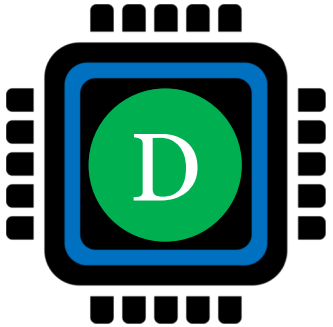


Parallel Tasks

III



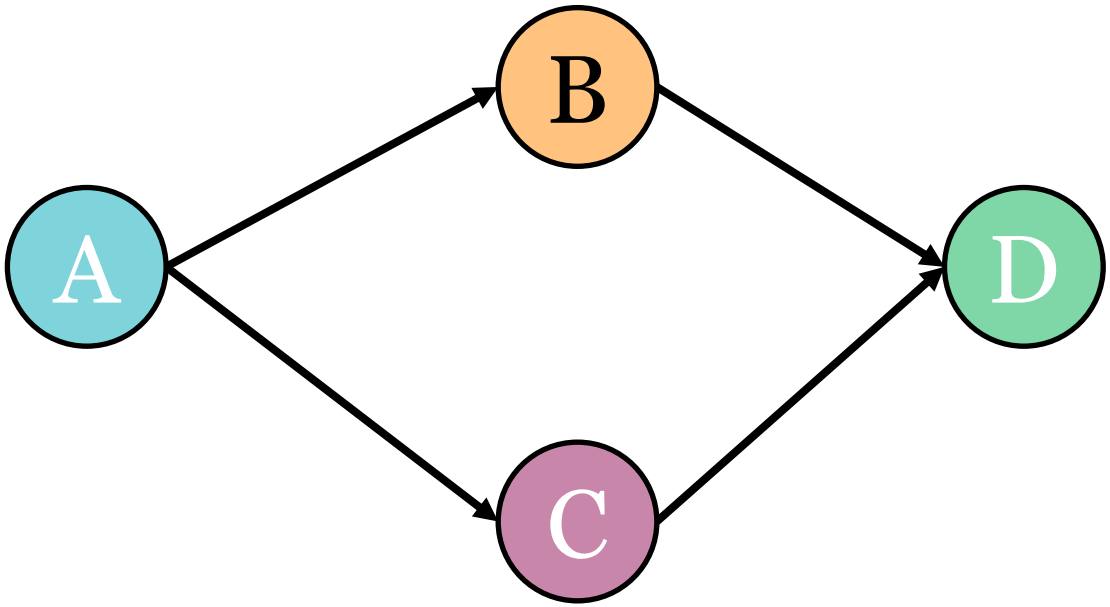


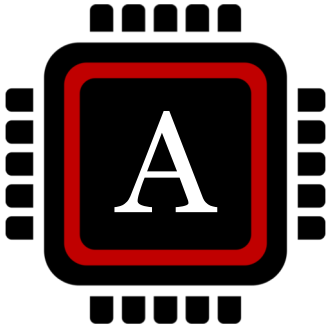


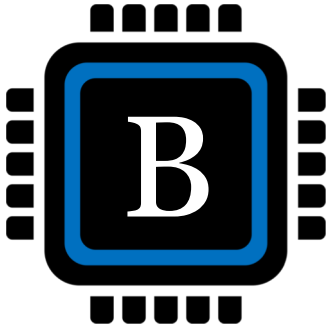
Time = 3

Parallel Tasks

III



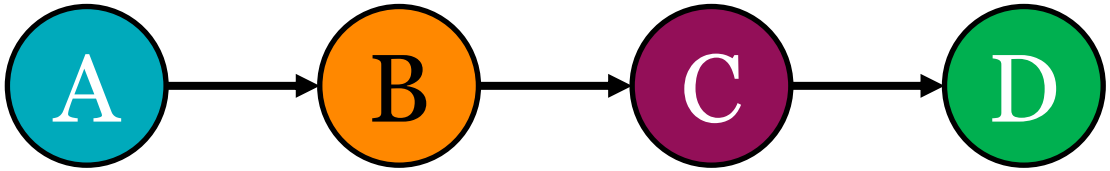


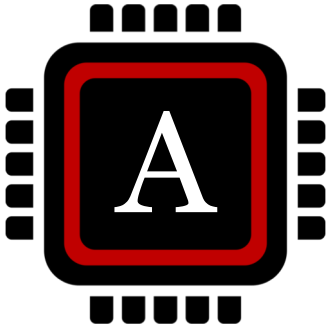


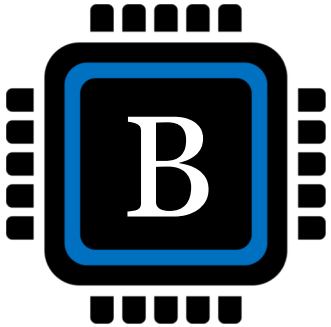
Time = 4

Sequential Task

VI

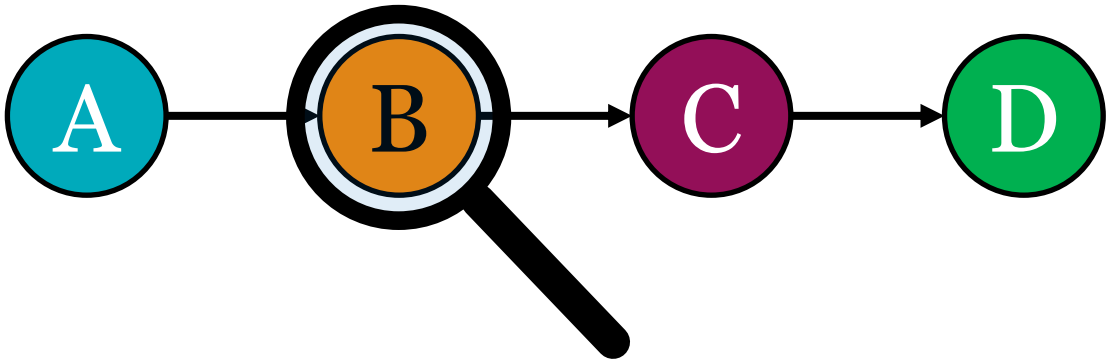


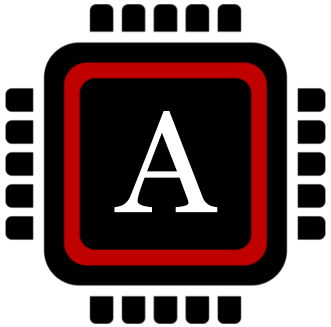


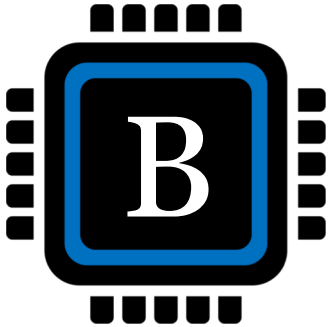


Sequential Task

VI

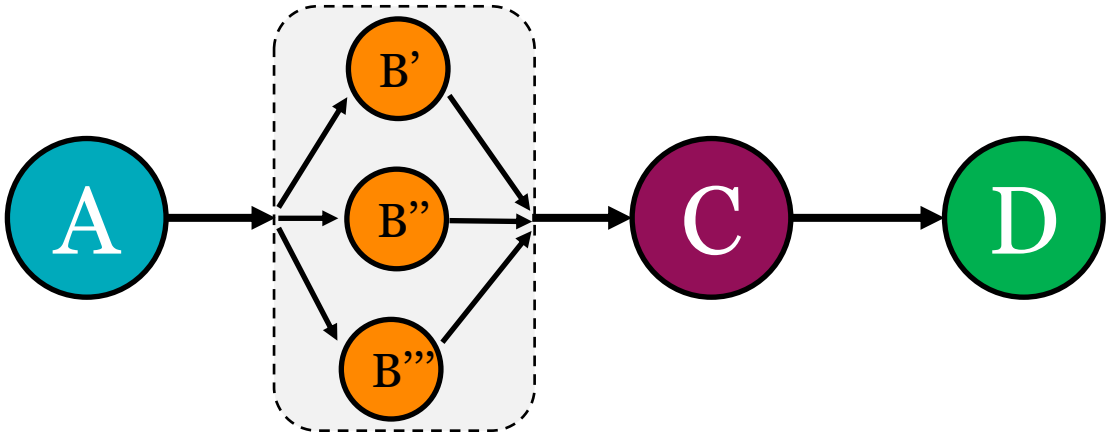


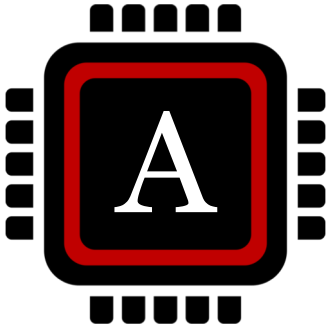


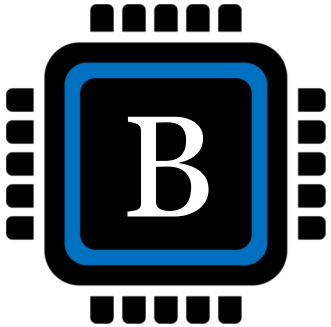


Sequential Task

VI

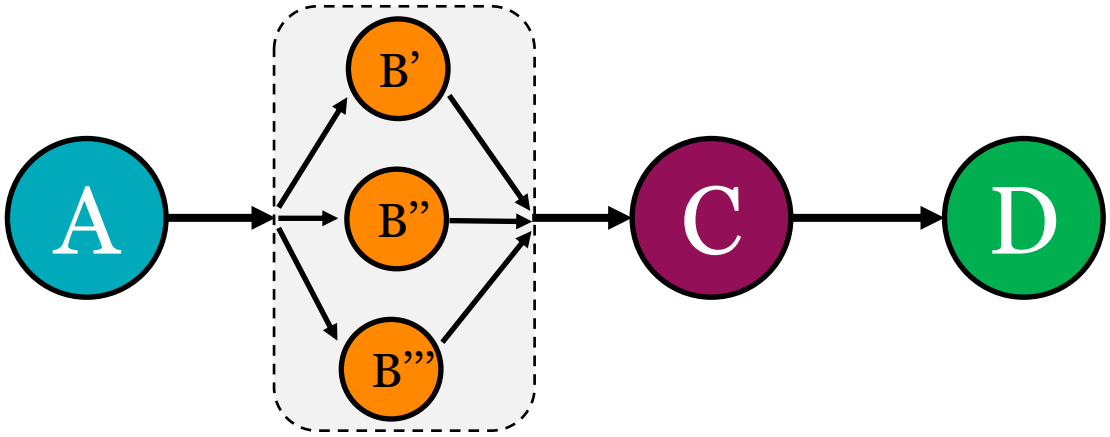




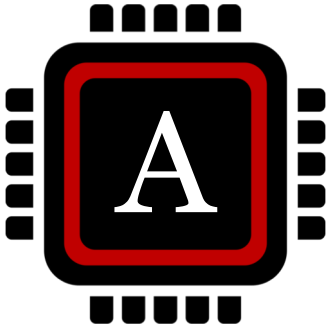


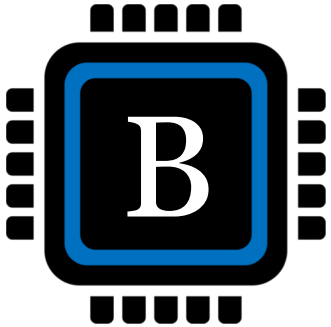
Sequential Task

VI



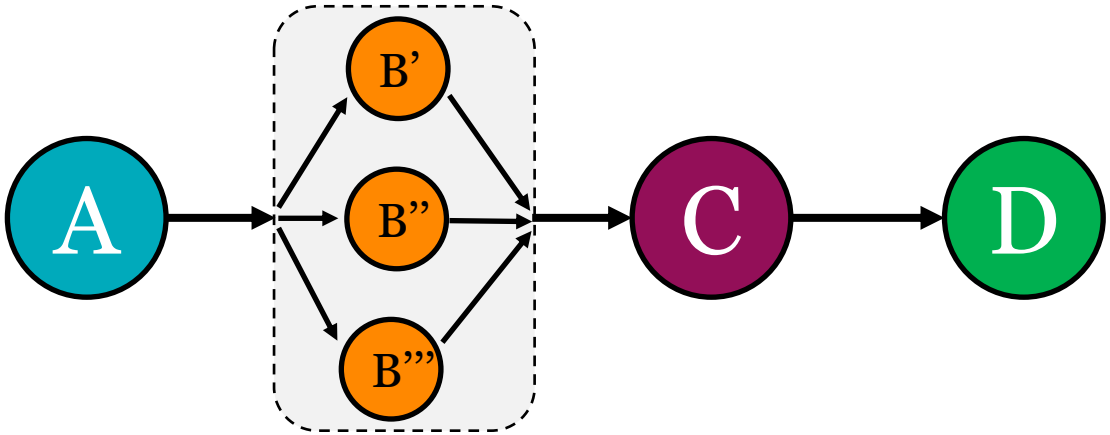
Same tasks, working on different parts of the data

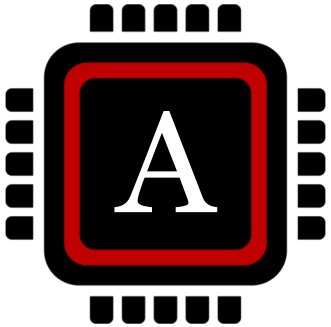


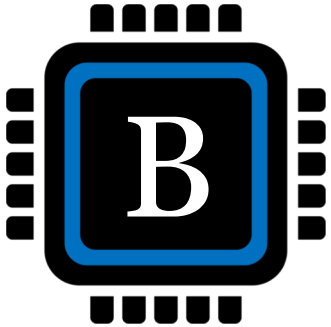


Sequential Task

VI

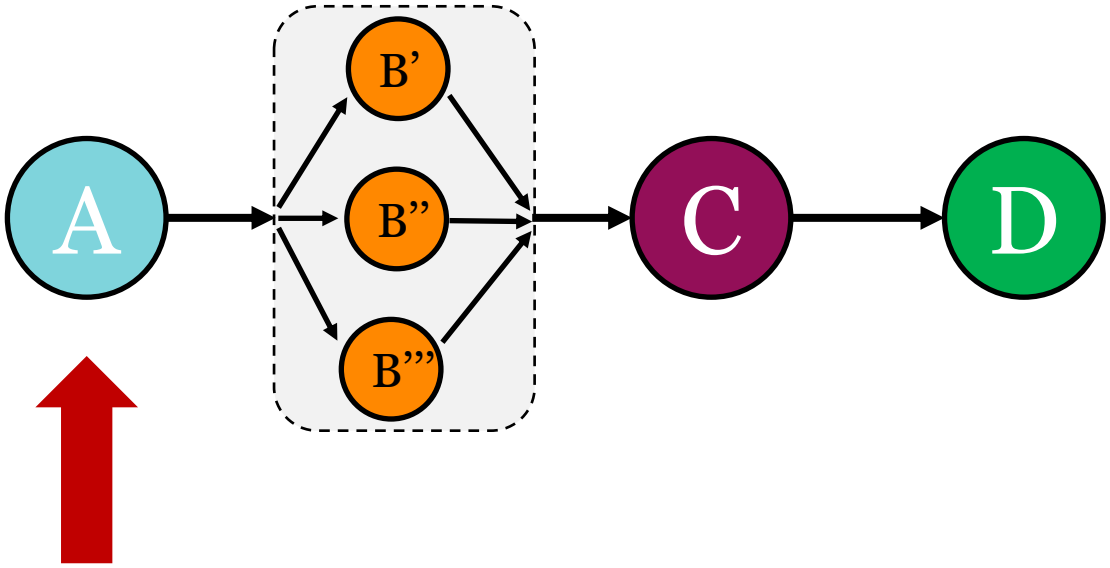


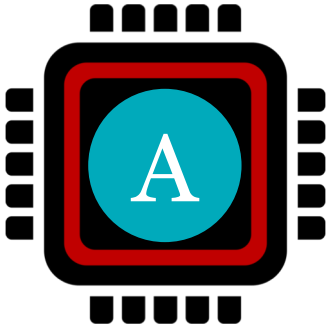


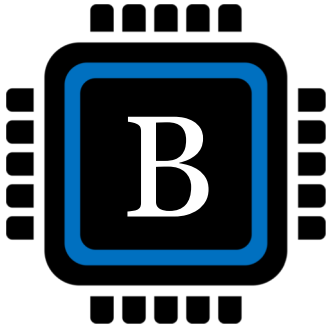


Sequential Task

VI



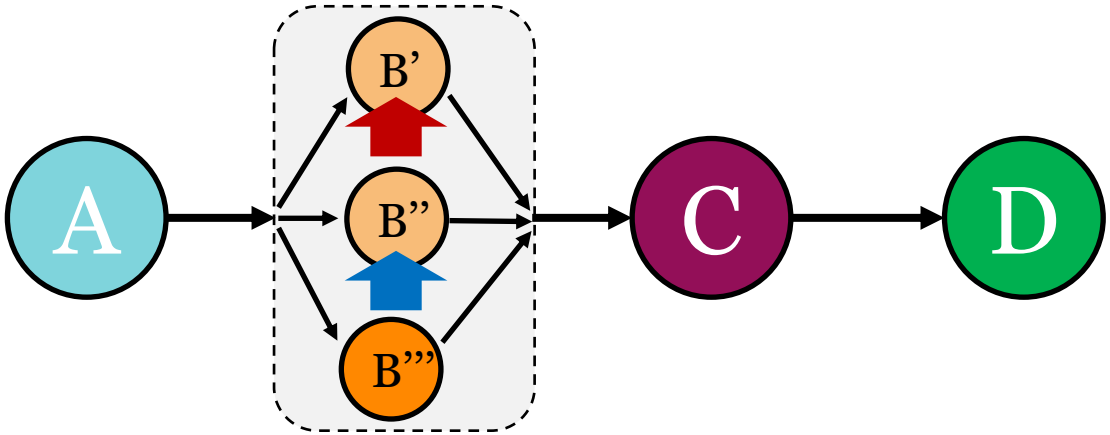


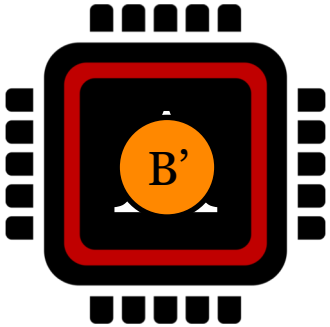


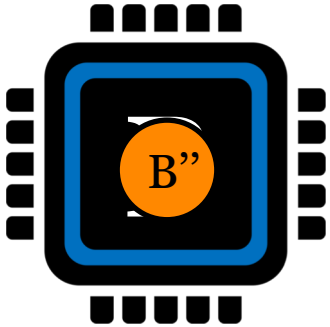
Time = 1

Sequential Task

VI



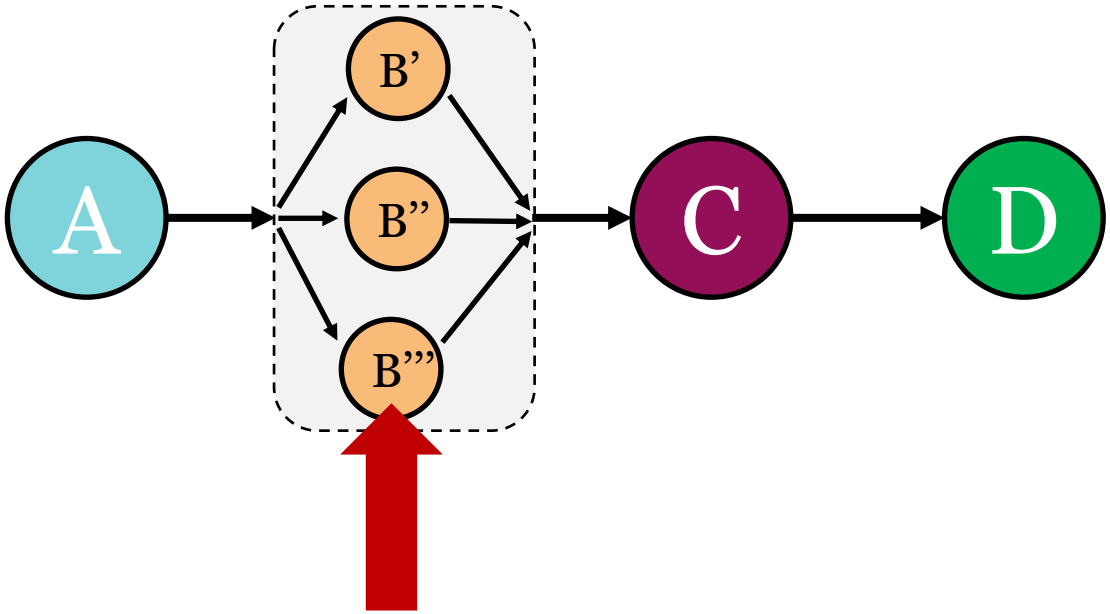


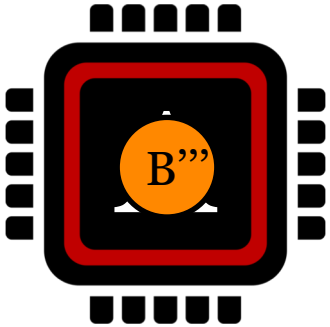


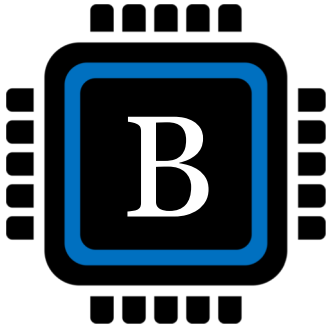
Time = 2

Sequential Task

VI



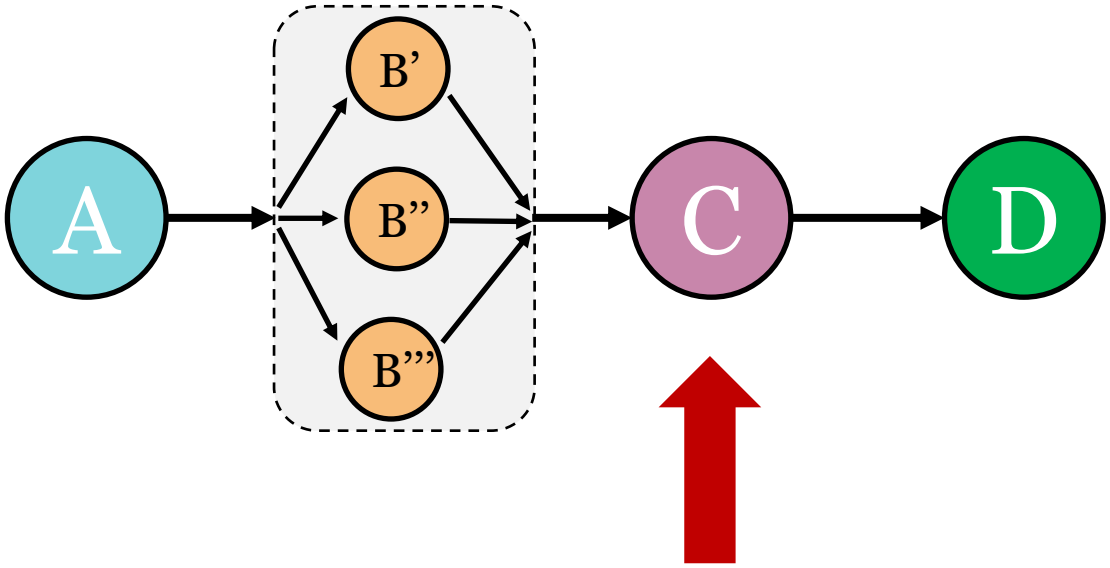


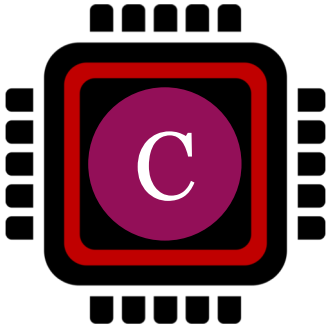


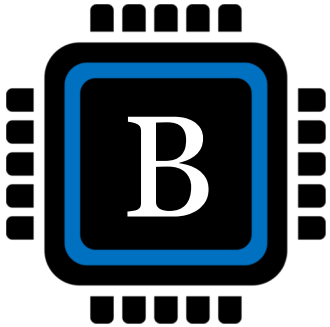
Time = 3

Sequential Task

VI



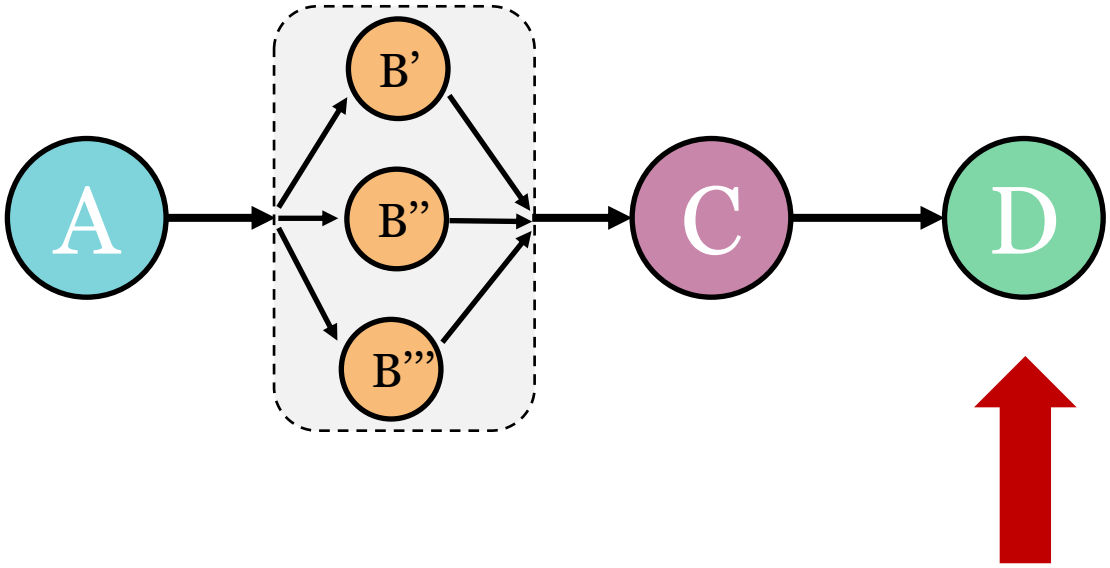


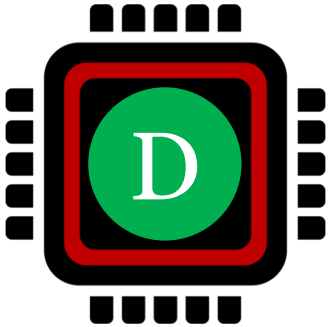


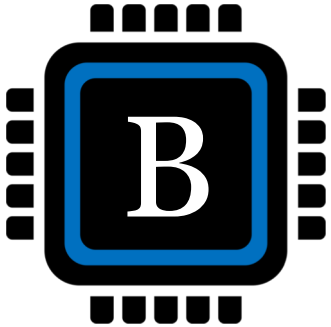
Time = 4

Sequential Task

VI



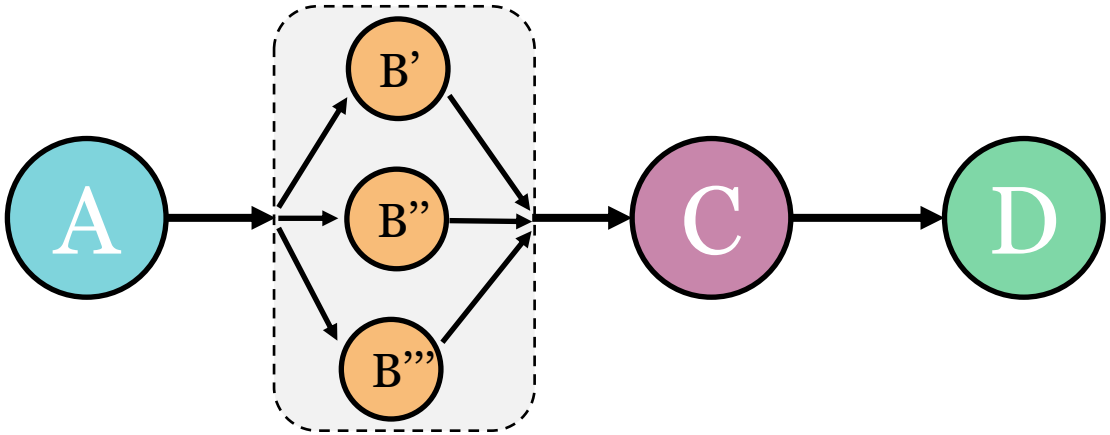


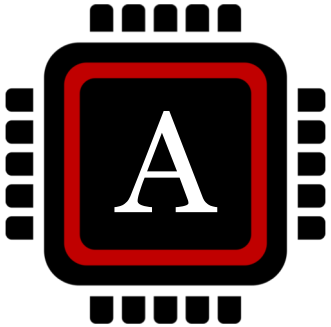


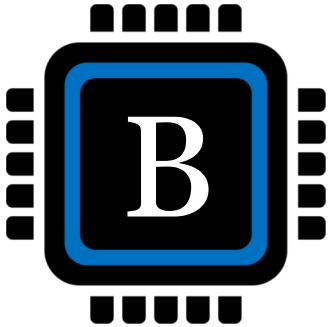
Time = 5

Sequential Task

VI







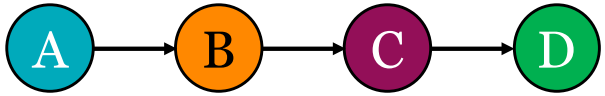
Time = 6

Workload Overview

I

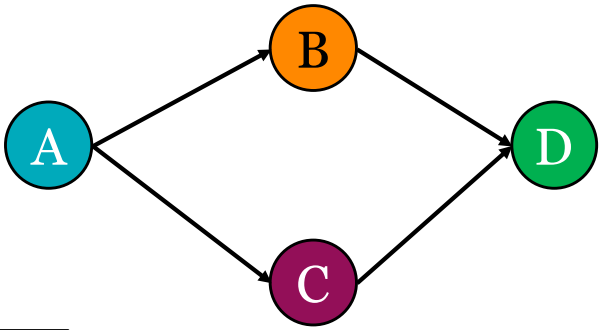
Time = 2

II



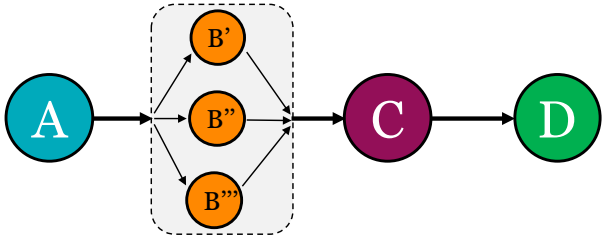
Time = 4

III



Time = 3

VI



Time = 5

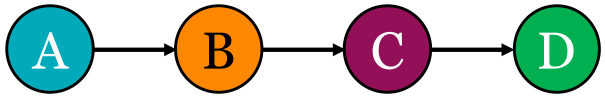
Workload Overview

Which type of application fits into which category?

I

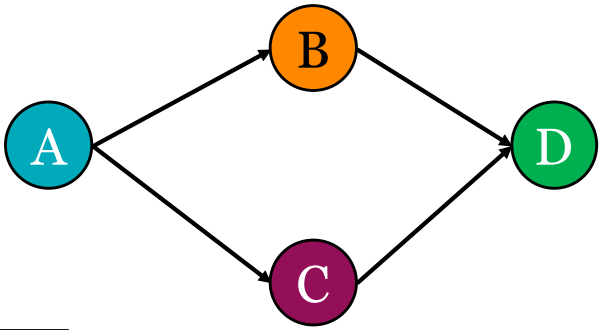
Time = 2

II



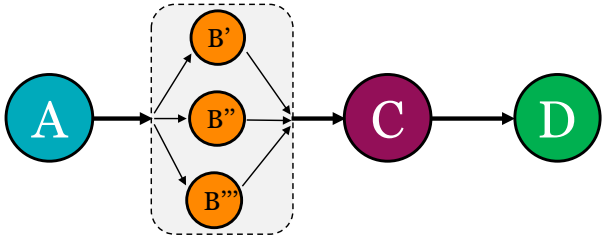
Time = 4

III



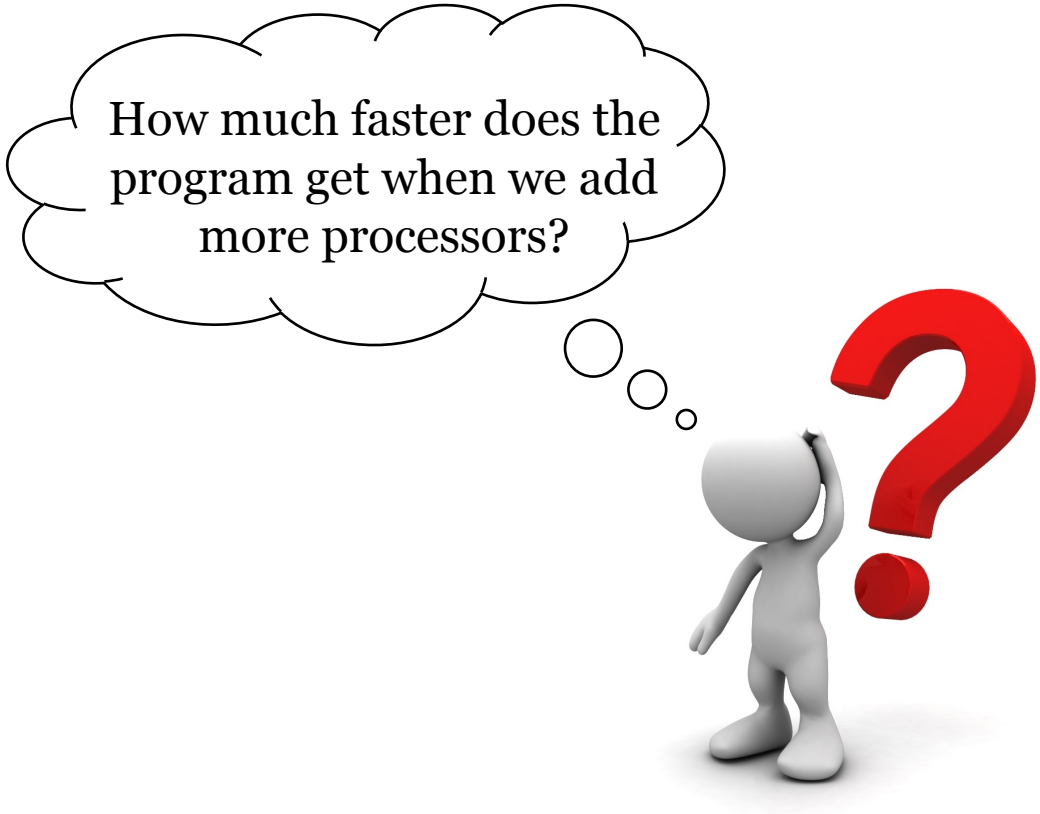
Time = 3

VI

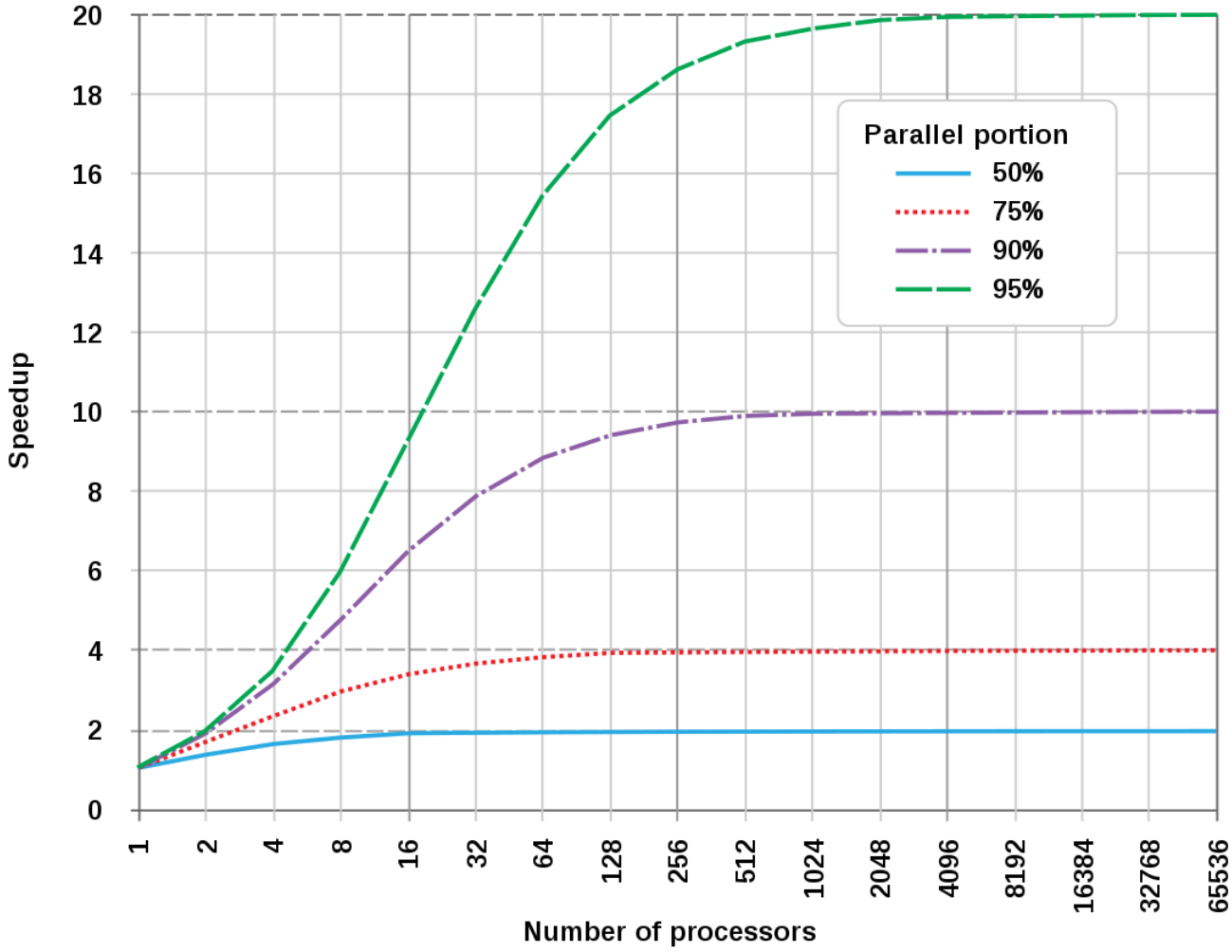


Time = 5

Amdahl's Law

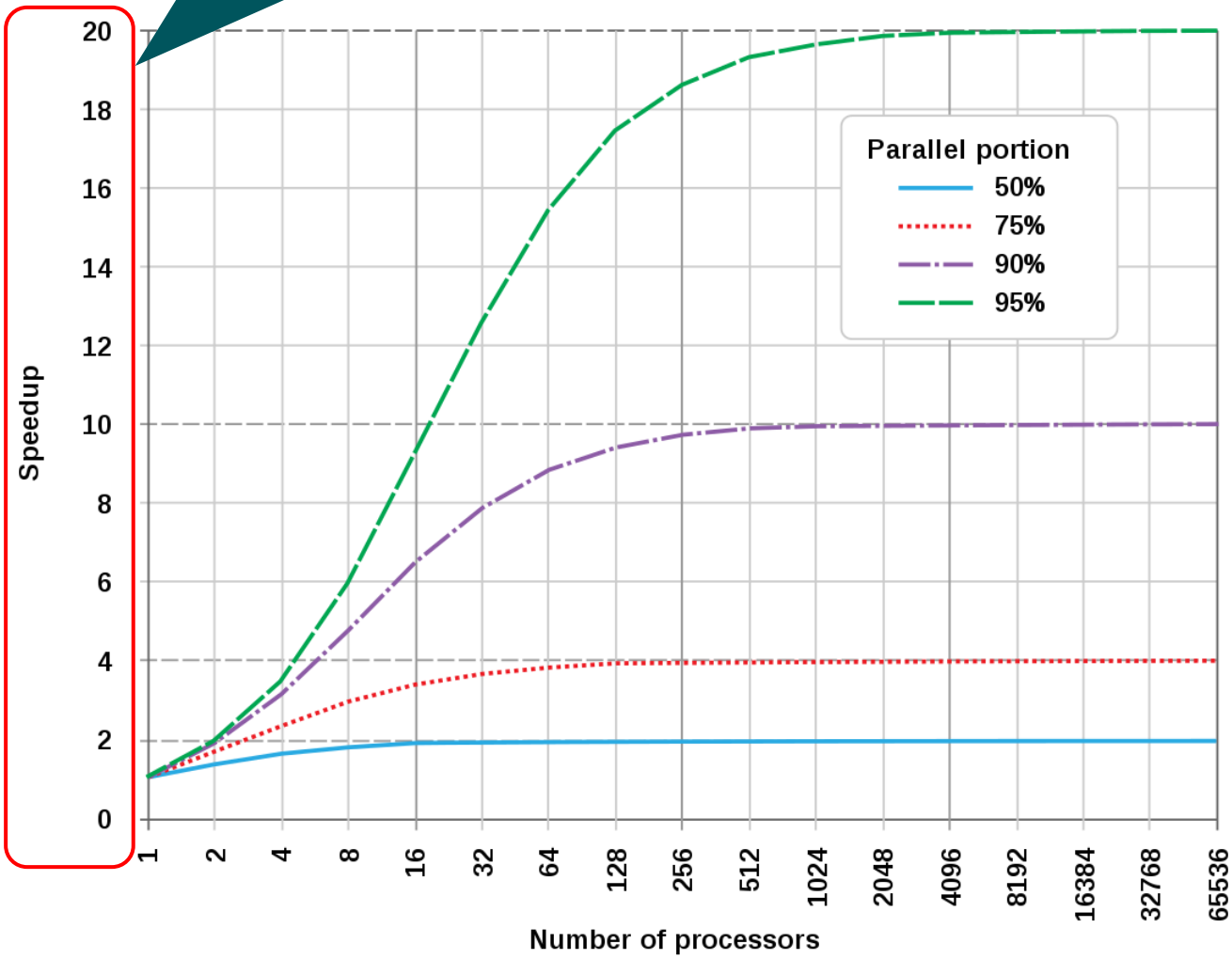

How much faster does the program get when we add more processors?

Amdahl's Law

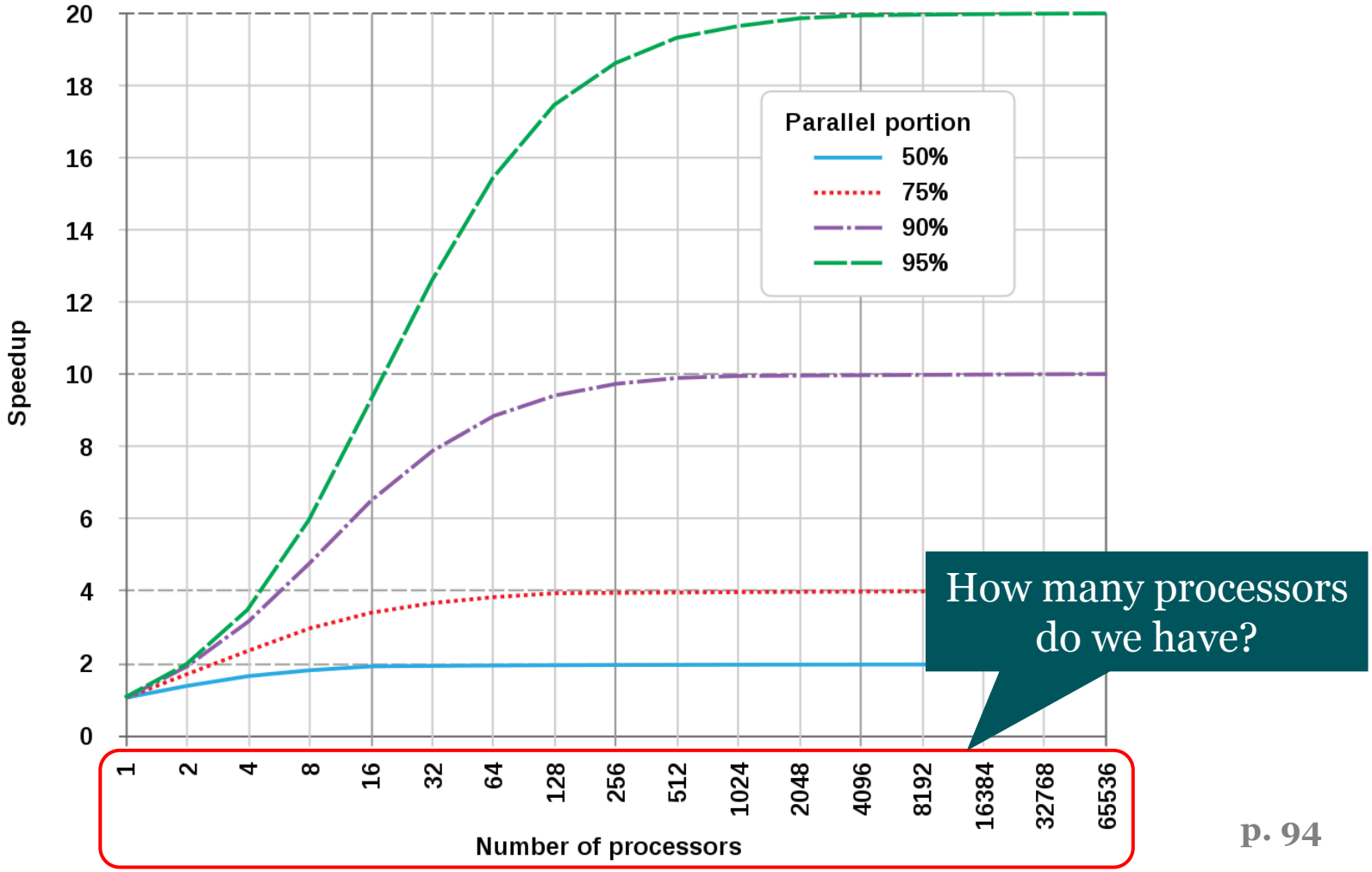

Speedup
Number of processors
Parallel portion
50%
75%
90%
95%
0
2
4
6
8
10
12
14
16
18
20
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536

Amdahl's Law

Theoretical speedup of the application

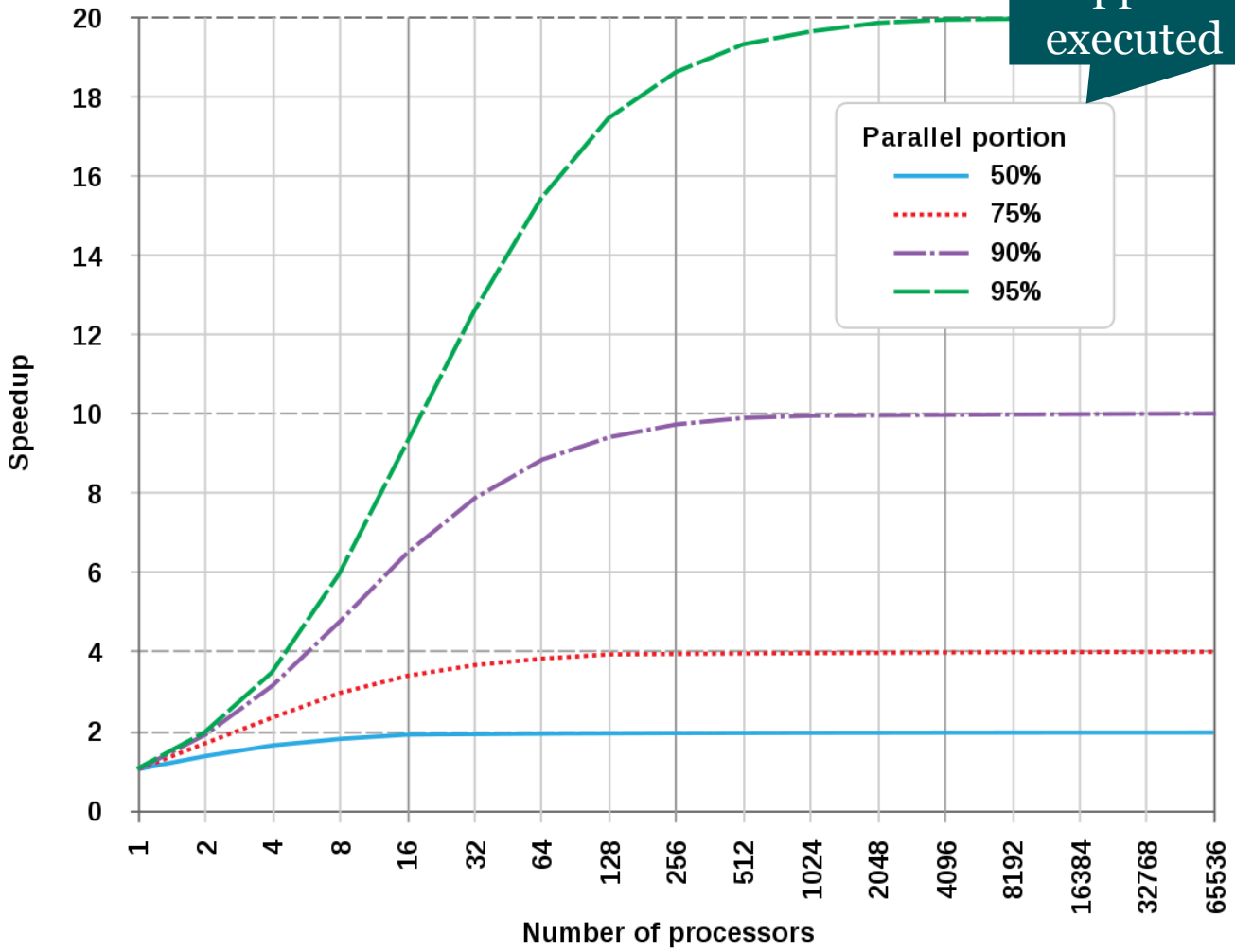


Amdahl's Law

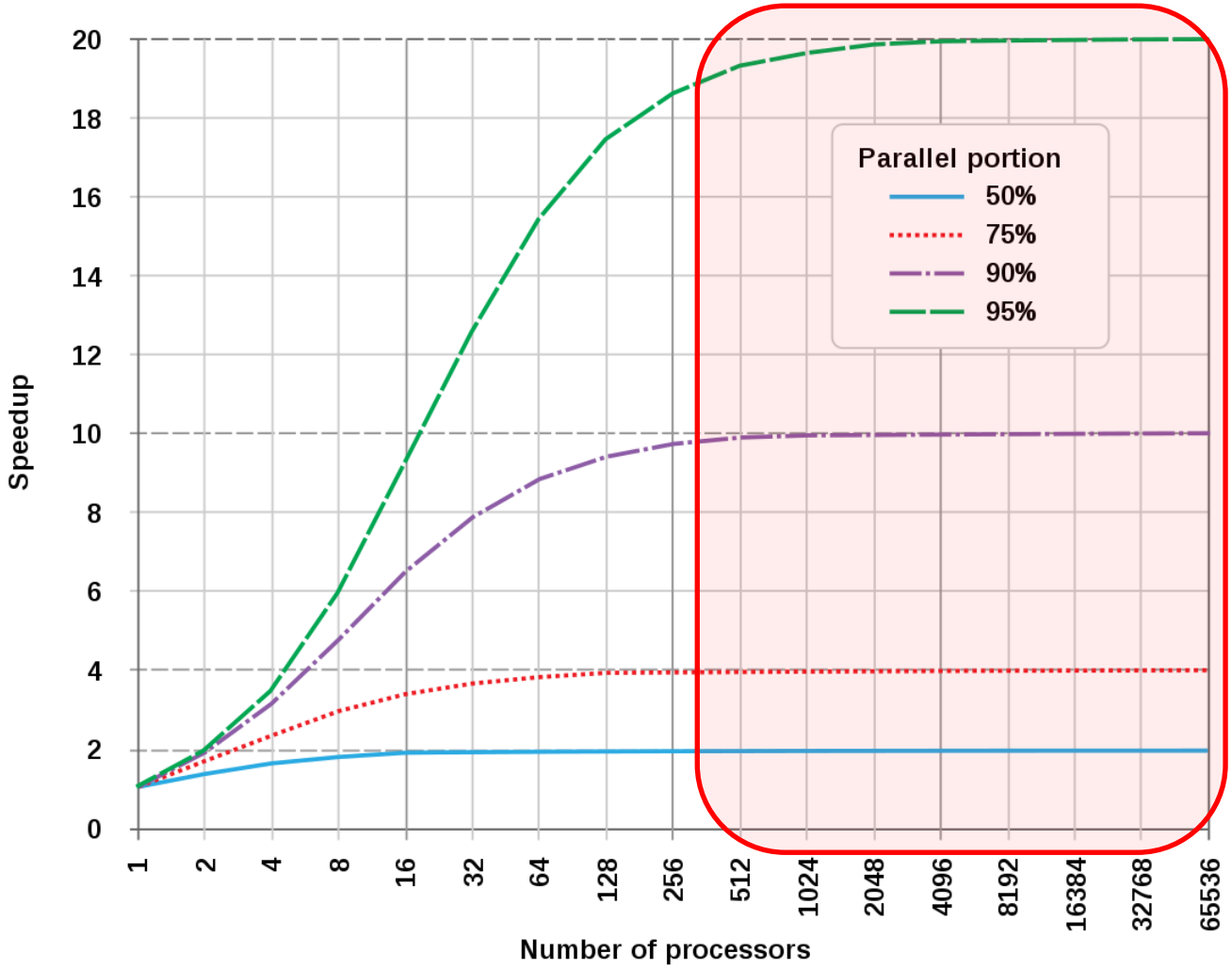

Speedup
20
18
16
14
12
10
8
6
4
2
0
Parallel portion
50%
75%
90%
95%
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
Number of processors
How many processors do we have?

Amdahl's Law

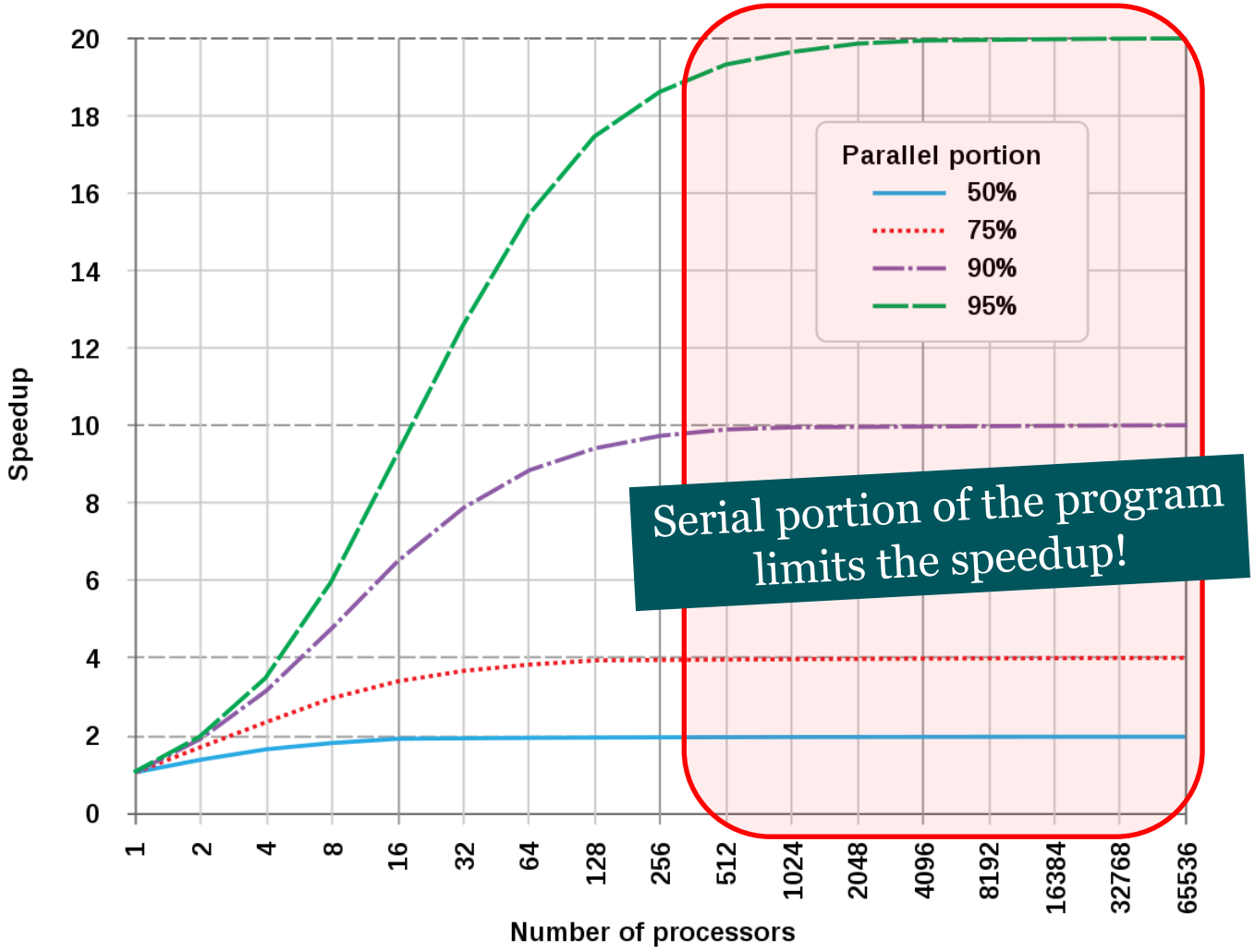
How much of the application can be executed in parallel?



Amdahl's Law

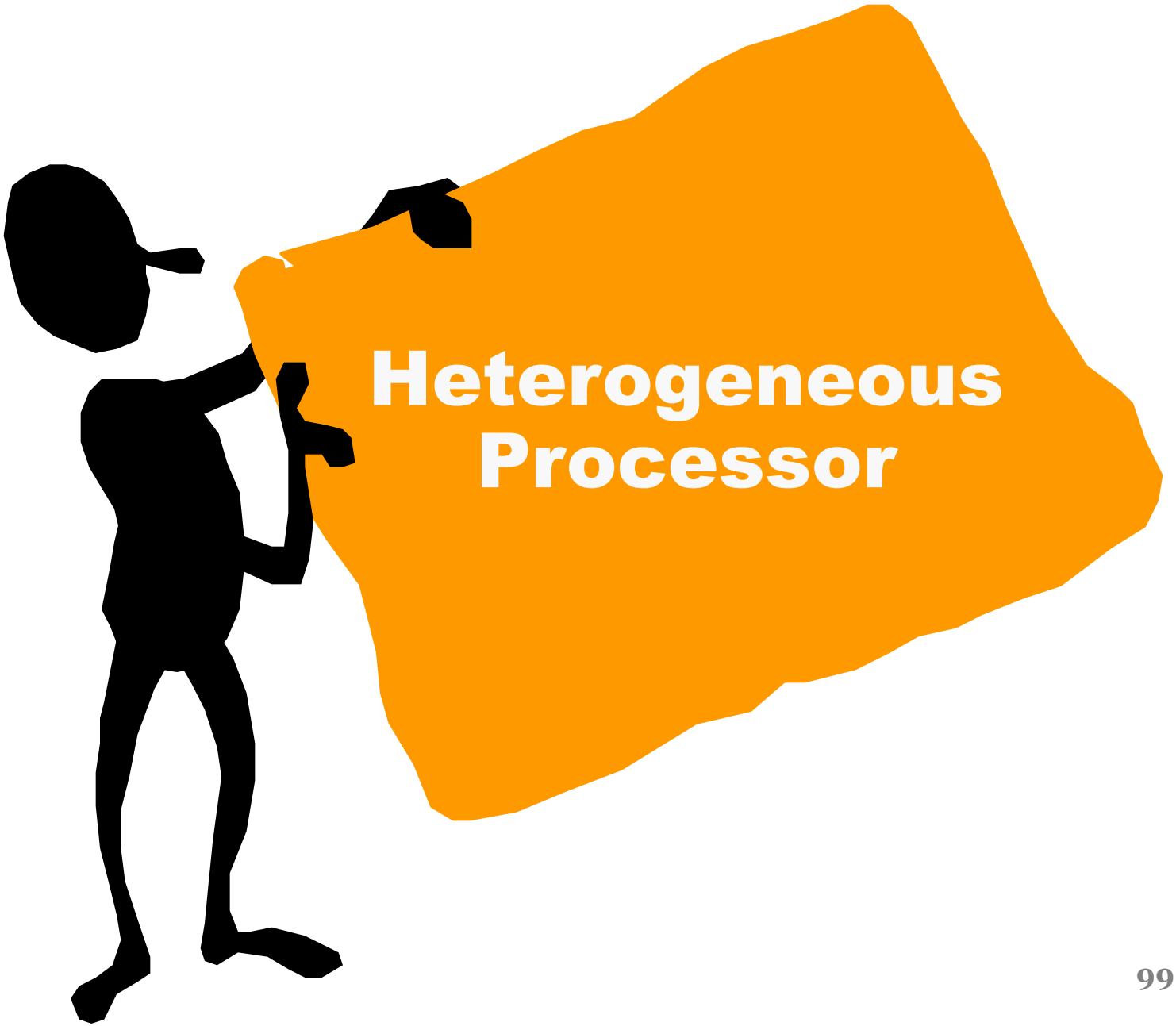

Speedup
20
18
16
14
12
10
8
6
4
2
0
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
Number of processors
Parallel portion
50%
75%
90%
95%

Amdahl's Law

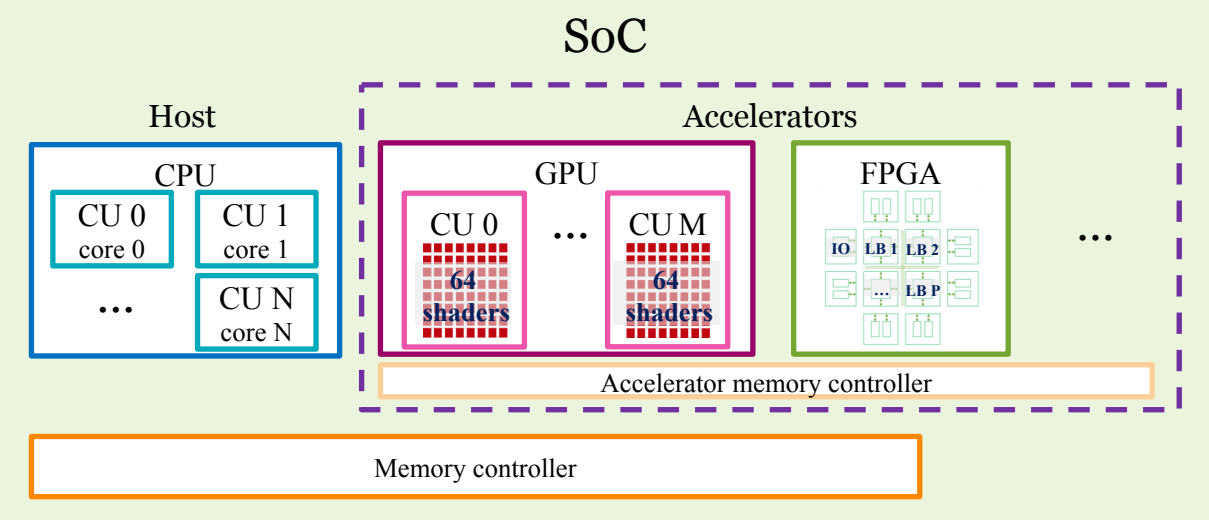

Speedup
20
18
16
14
12
10
8
6
4
2
0
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
Number of processors
Parallel portion
50%
75%
90%
95%
Serial portion of the program limits the speedup!

Question

- What are serial parts in a program that always exist?

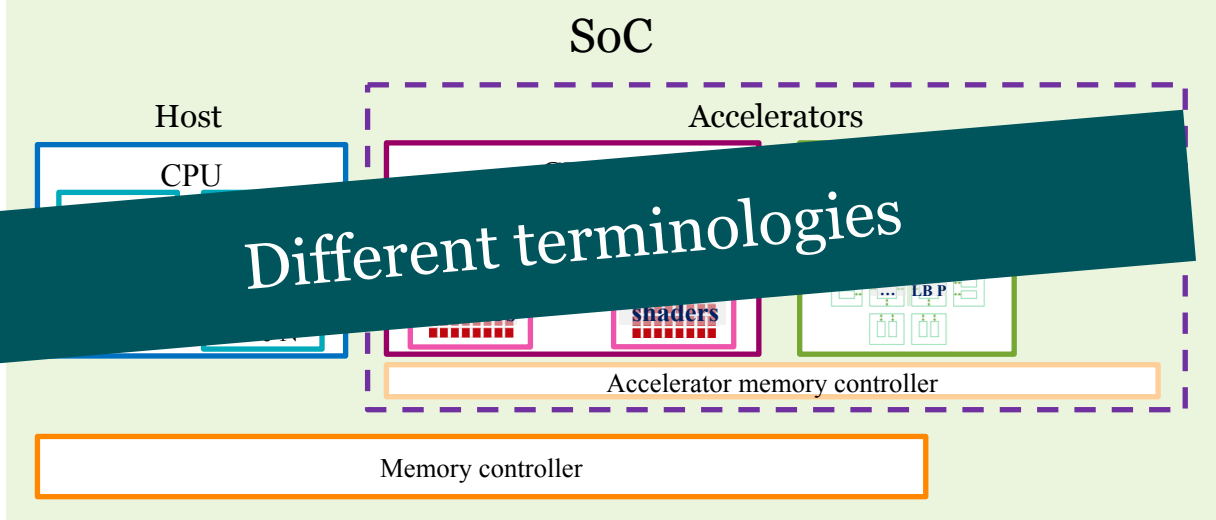

Heterogeneous
Processor

Heterogeneous Processor



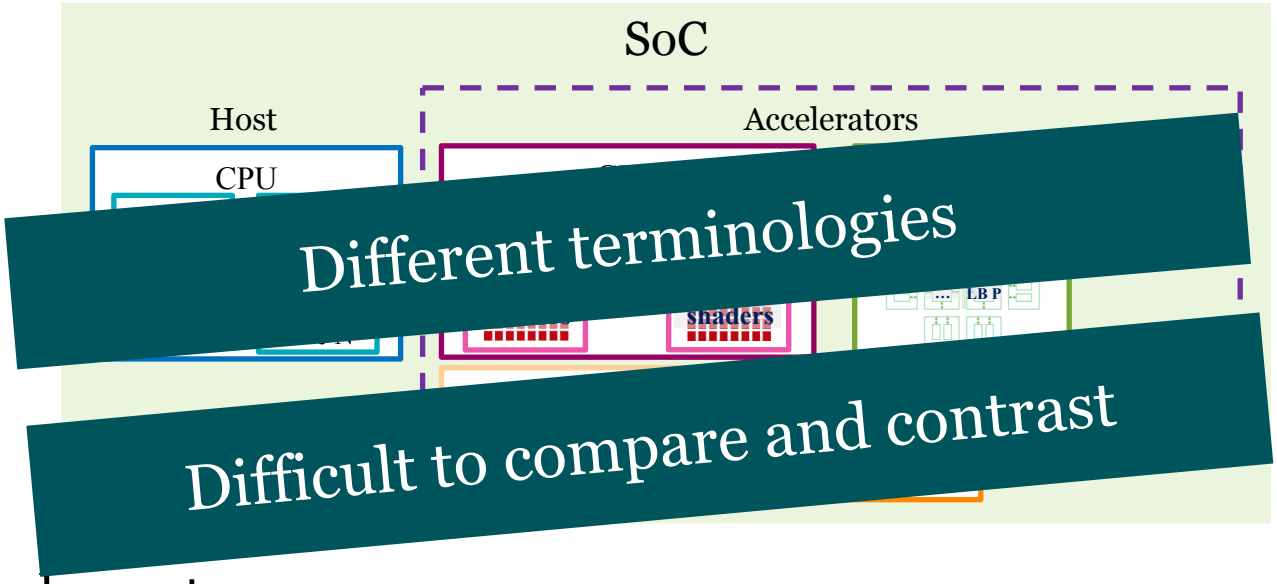
- Main elements:
  - Compute Units & Logic Blocks (LBs) in PUs / Shaders & Cores
  - Interconnection between external memory and PUs
  - Memory models

Heterogeneous Processor



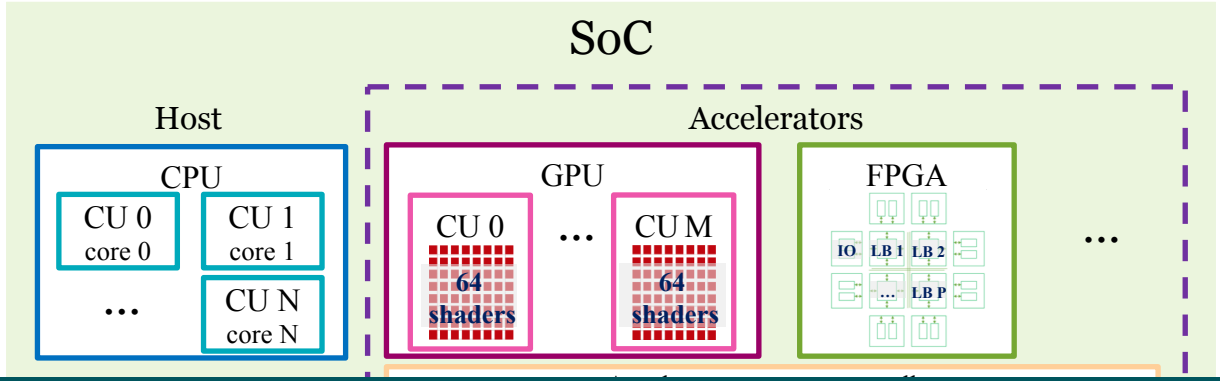
- Main elements:
  - Compute Units & Logic Blocks (LBs) in PUs / Shaders & Cores
  - Interconnection between external memory and PUs
  - Memory models

Heterogeneous Processor



- Main elements:
  - Compute Units & Logic Blocks (LBs) in PUs / Shaders & Cores
  - Interconnection between external memory and PUs
  - Memory models

Heterogeneous Processor

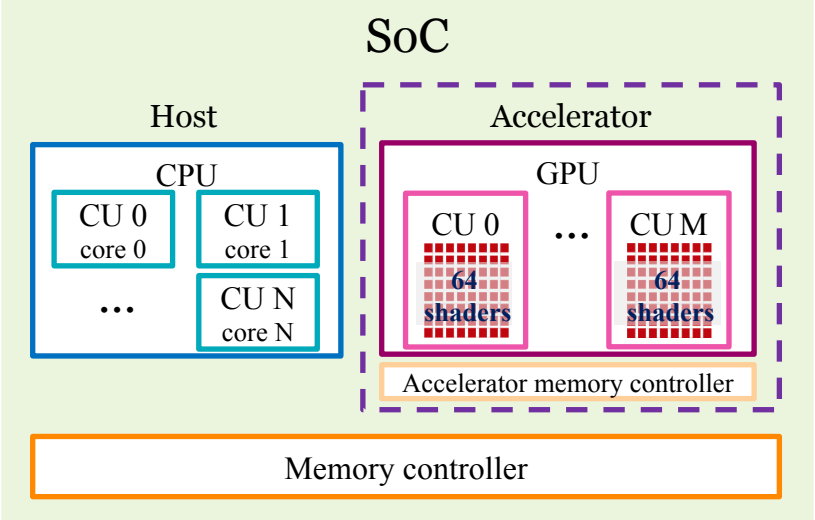


PU: Processing Unit;
CPU, GPU, FPGA and etc

CU: Compute Unit;
Cores in CPU, CUs in GPU, Logic Blocks in FPGA and etc

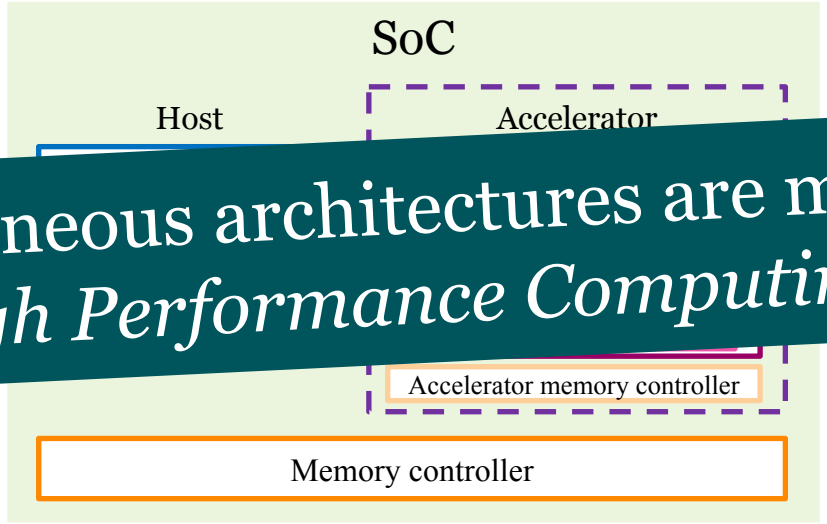
- Memory models

CPU + GPU



- Main elements:
  - Compute Units in PUs / Shaders & Cores
  - Interconnection between external memory and PUs
  - Memory models

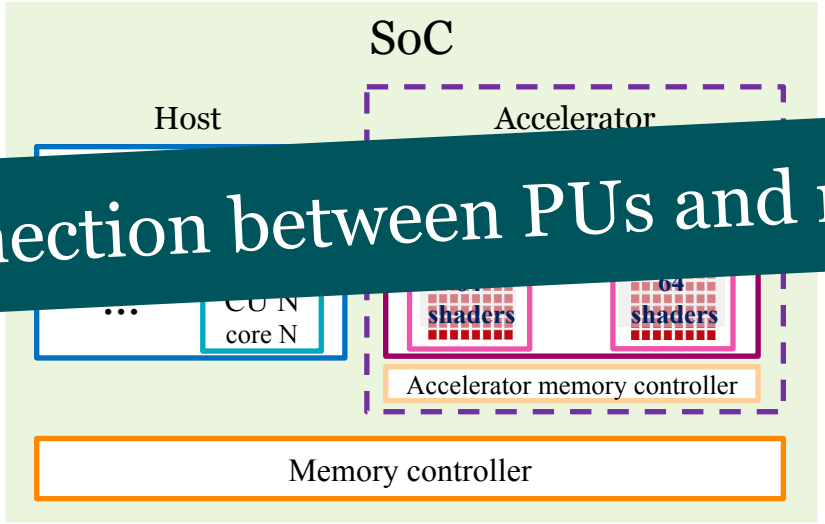
CPU + GPU



Heterogeneous architectures are made for *High Performance Computing*

- Main elements:
  - Compute Units in PUs / Shaders & Cores
  - Interconnection between external memory and PUs
  - Memory models

CPU + GPU

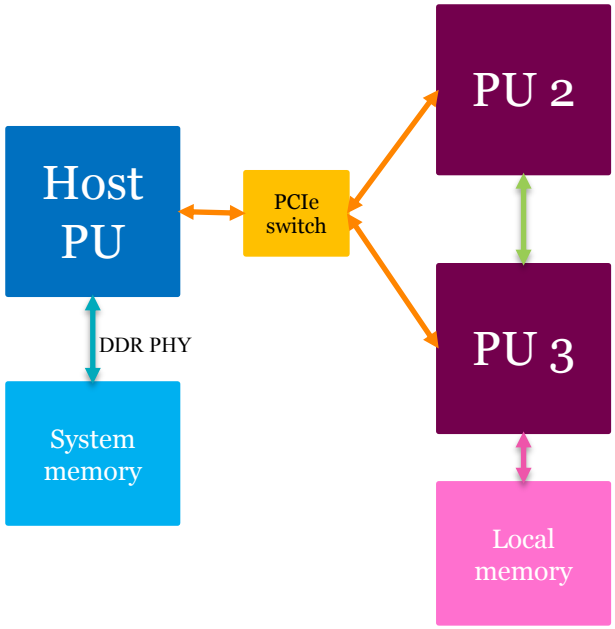


Interconnection between PUs and memory

- Main elements:
  - Compute Units in PUs / Shaders & Cores
  - Interconnection between external memory and PUs
  - Memory models

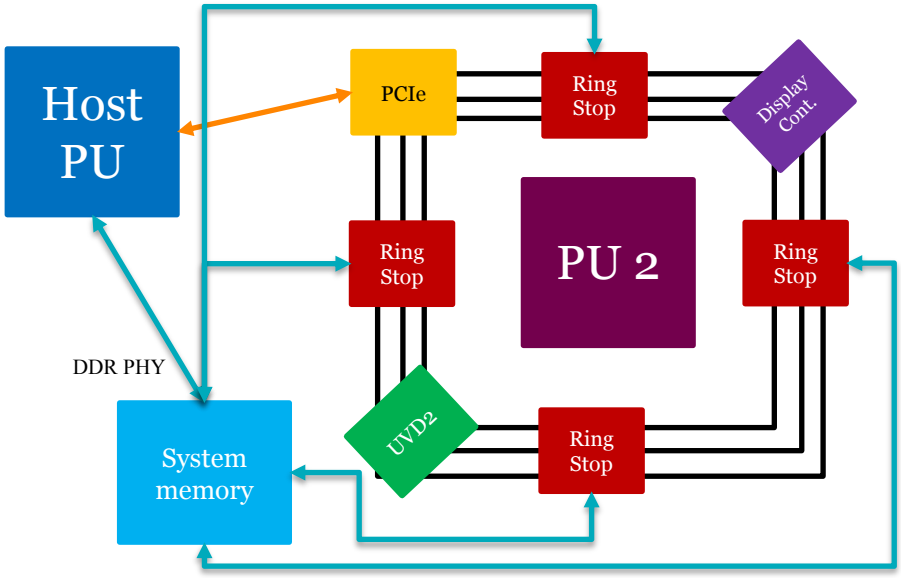
Interconnect, example 1

Heterogeneous processors



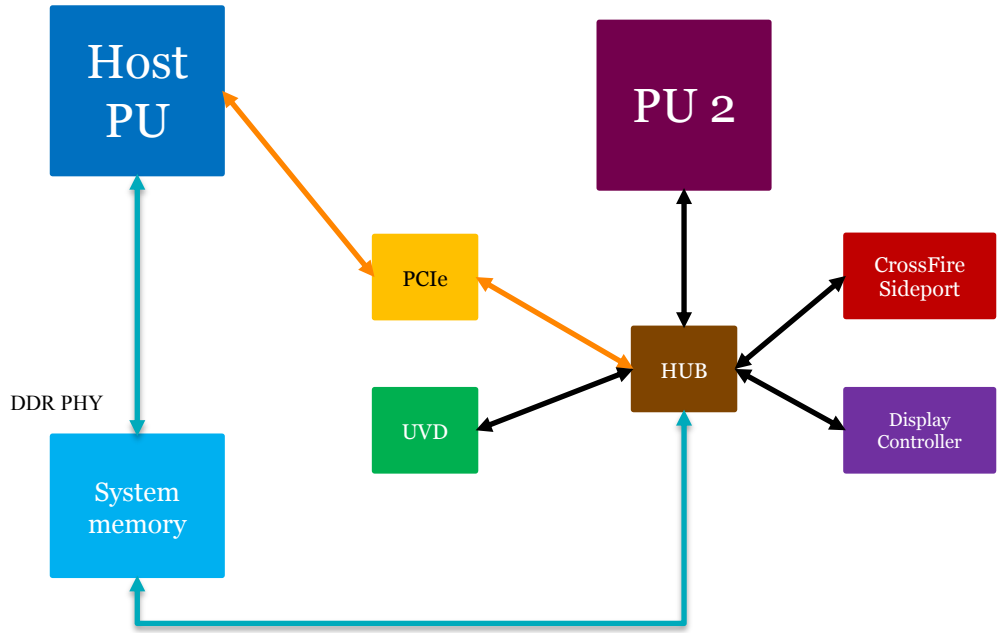
Interconnect, example 2

Heterogeneous processors



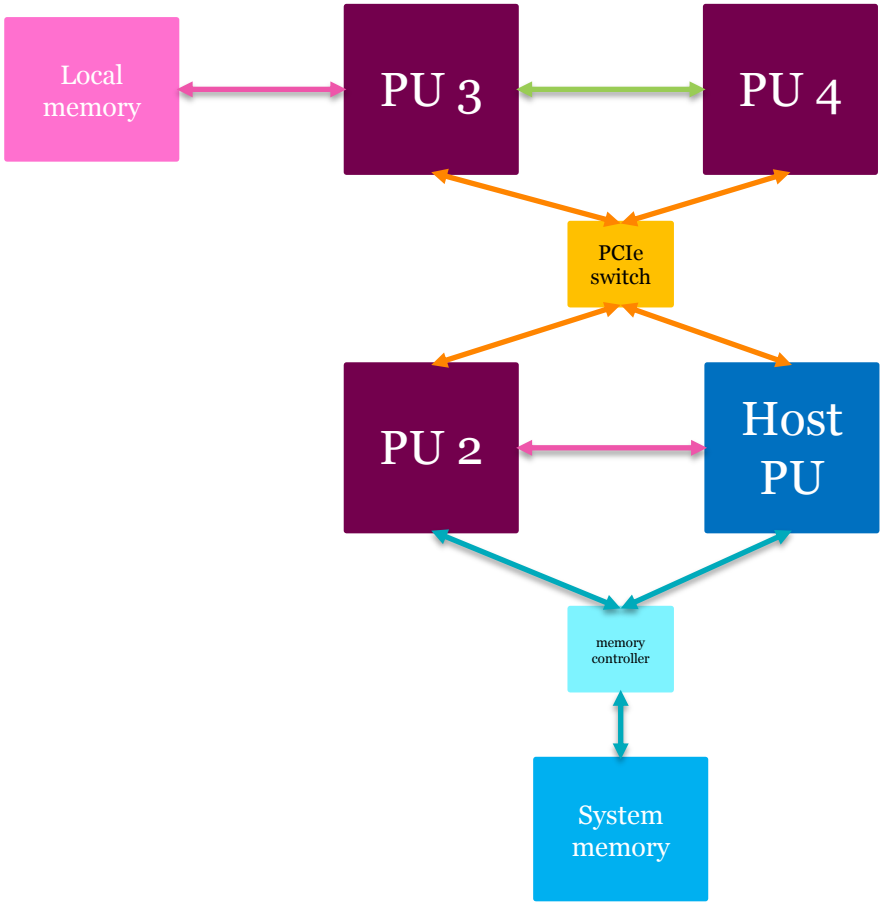
Interconnect, example 3

Heterogeneous processors



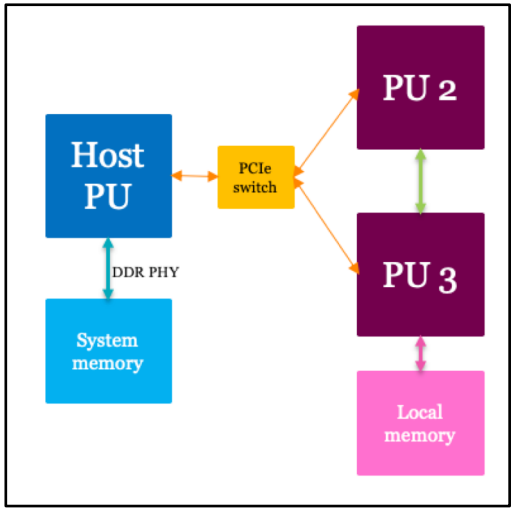
Interconnect, example 4

Heterogeneous processors



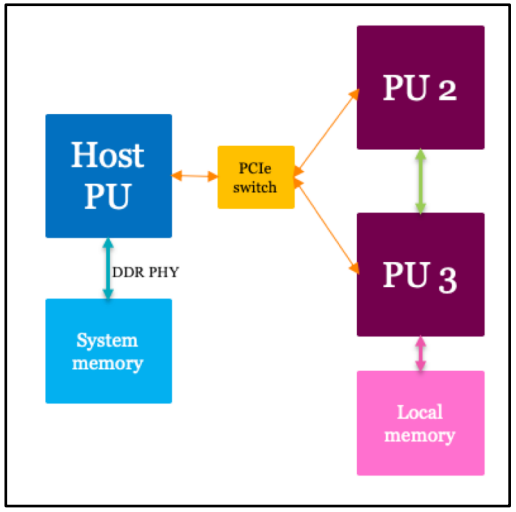
Interconnect

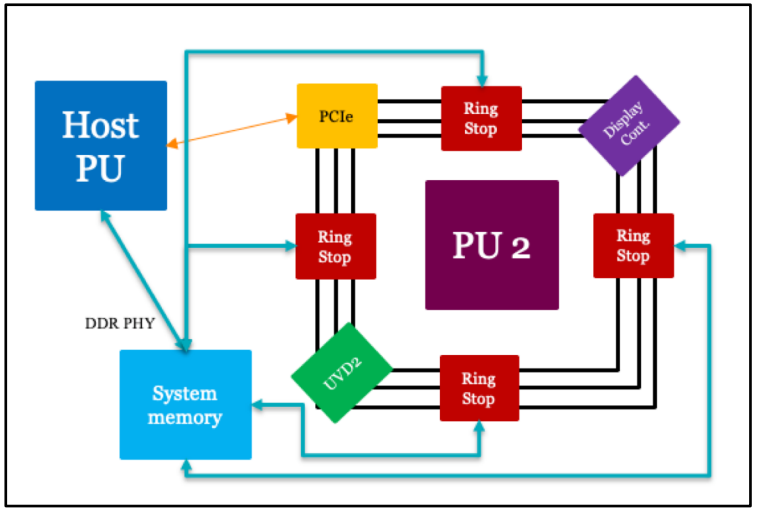
Heterogeneous processors



Interconnect

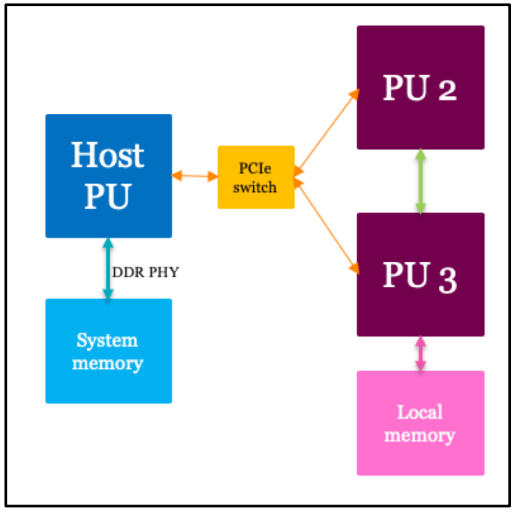
Heterogeneous processors

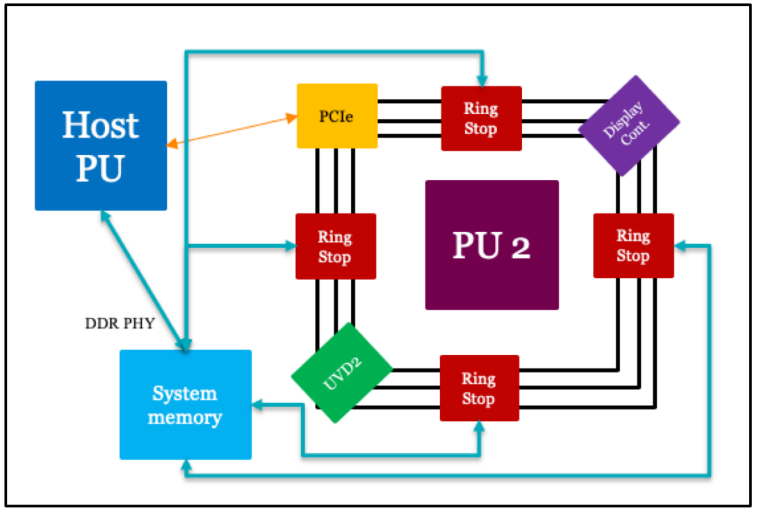


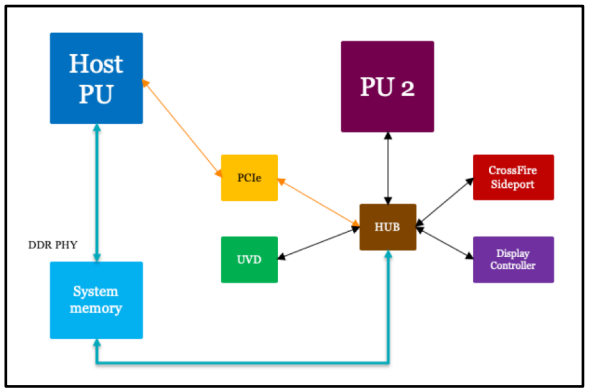


Interconnect

Heterogeneous processors

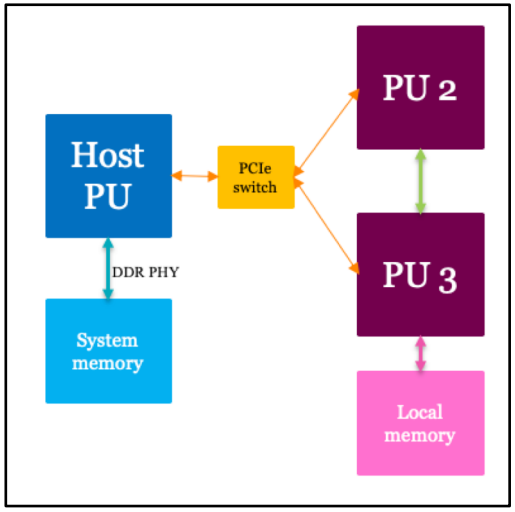


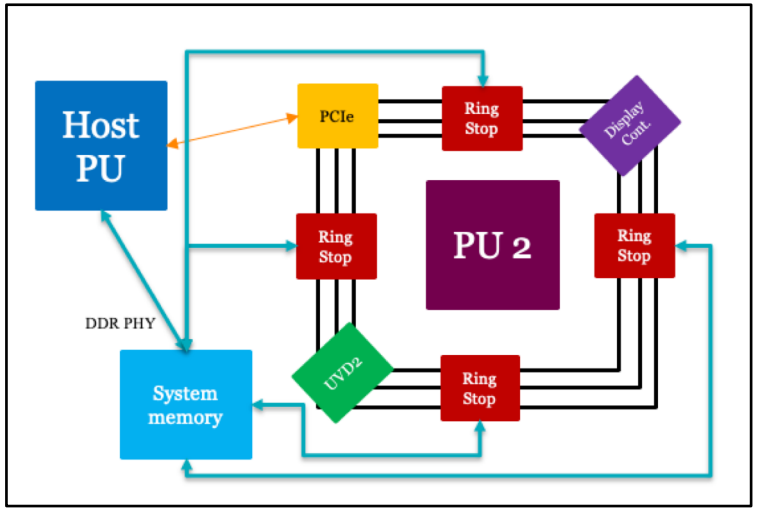


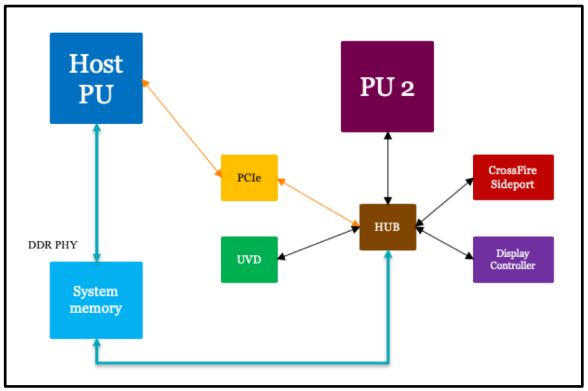


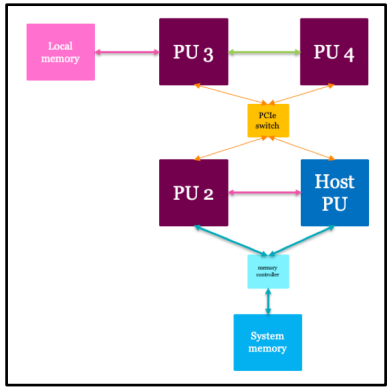
Interconnect

Heterogeneous processors



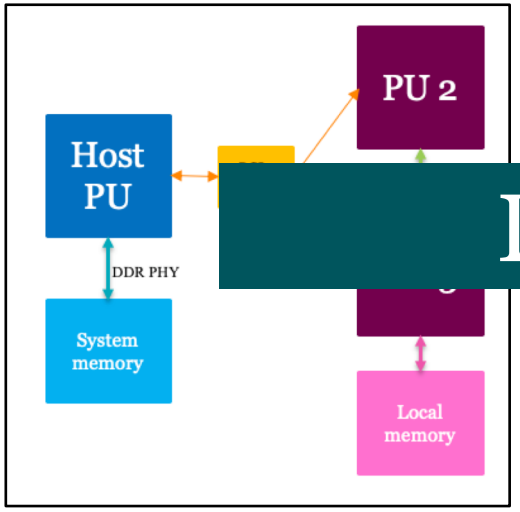


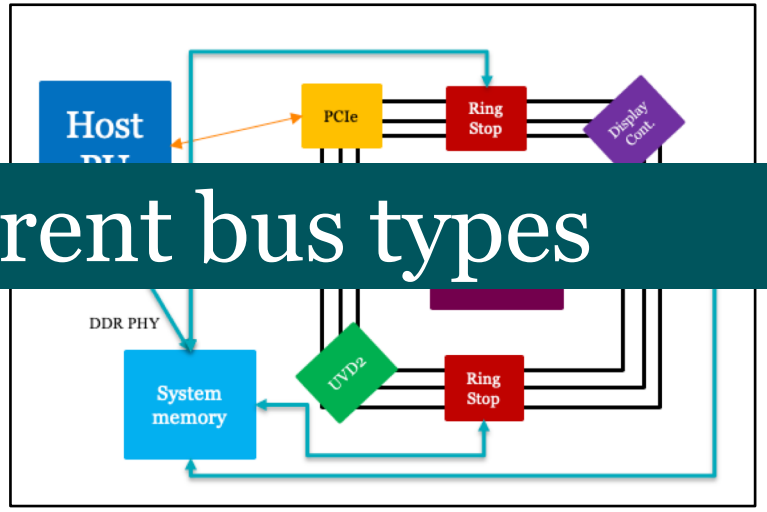




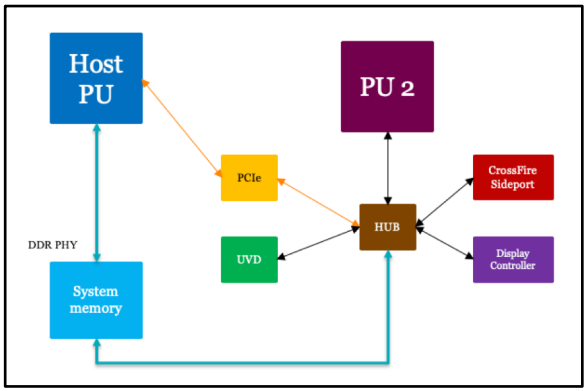
Interconnect

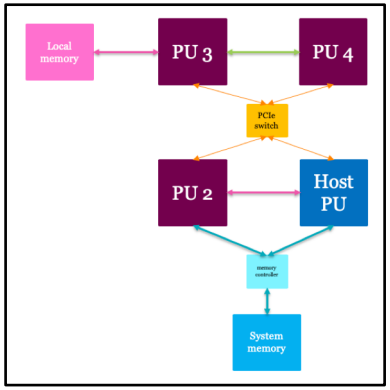
Heterogeneous processors





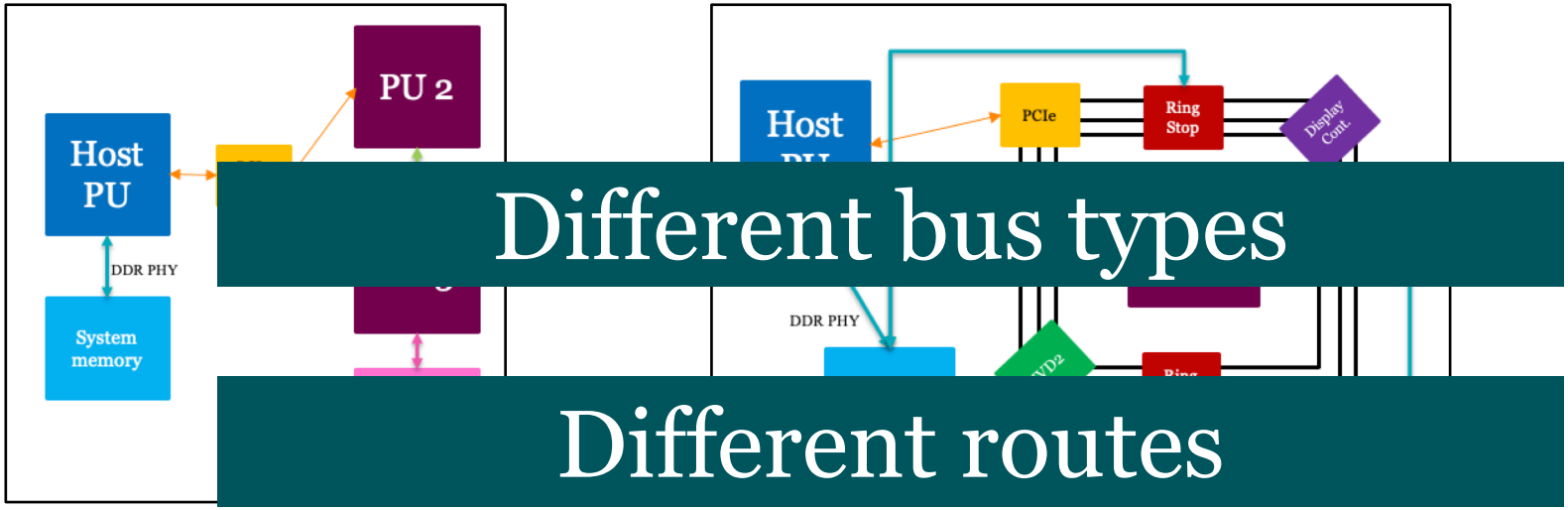
Different bus types

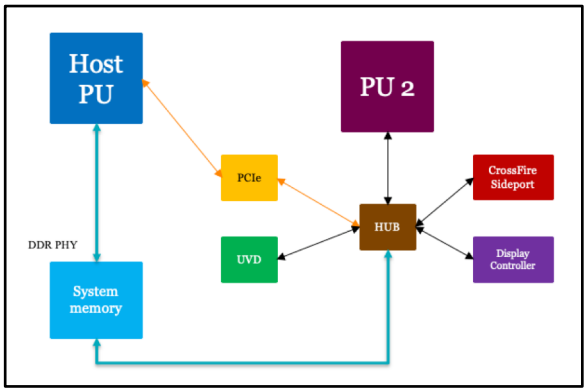


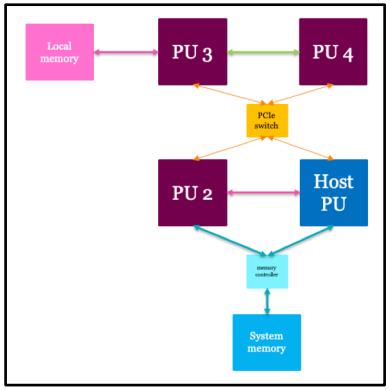


Interconnect

Heterogeneous processors

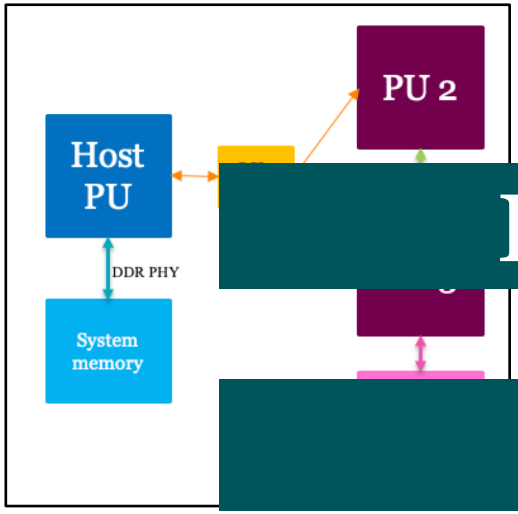


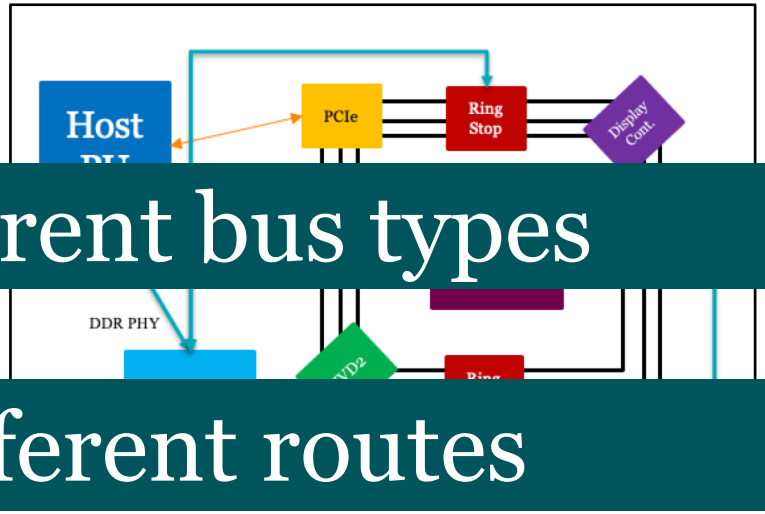




Interconnect

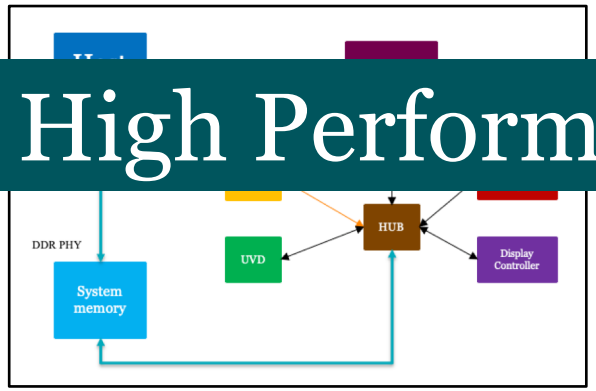
Heterogeneous processors

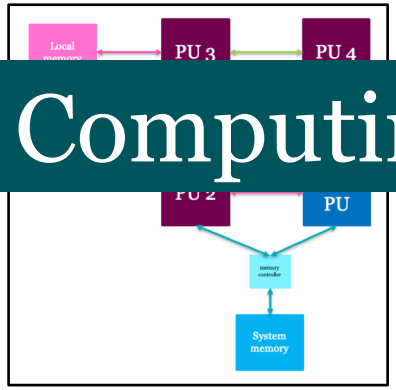




Different bus types

Different routes





High Performance Computing

Interconnect

Standards & specifications

- Three new bus/interconnect standards announced in 2016

Interconnect

Standards & specifications

- Three new bus/interconnect standards announced in 2016
  - Cache Coherent Interconnect for Accelerators (CCIX)

Interconnect

Standards & specifications

- Three new bus/interconnect standards announced in 2016
  - Cache Coherent Interconnect for Accelerators (CCIX)
  - Gen-Z

Interconnect

Standards & specifications

- Three new bus/interconnect standards announced in 2016
  - Cache Coherent Interconnect for Accelerators (CCIX)
  - Gen-Z
  - Open Coherent Accelerator Processor Interface (OpenCAPI)

https://www.openfabrics.org/images/eventpresos/2017presentations/213_CCIXGen-Z_BBenton.pdf

Interconnect

Standards & specifications

- Three new bus/interconnect standards announced in 2016
  - Cache Coherent Interconnect for Accelerators (CCIX)
  - Gen-Z
  - Open Coherent Accelerator Processor Interface (OpenCAPI)
- One more standard announced by Intel leading group in 2019

https://www.openfabrics.org/images/eventpresos/2017presentations/213_CCIXGen-Z_BBenton.pdf

Interconnect

Standards & specifications

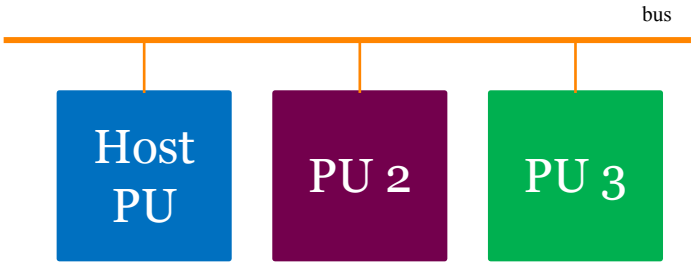
- Three new bus/interconnect standards announced in 2016
  - Cache Coherent Interconnect for Accelerators (CCIX)
  - Gen-Z
  - Open Coherent Accelerator Processor Interface (OpenCAPI)

- One more standard announced by Intel leading group in 2019
  - **Compute Express Link (CXL)**

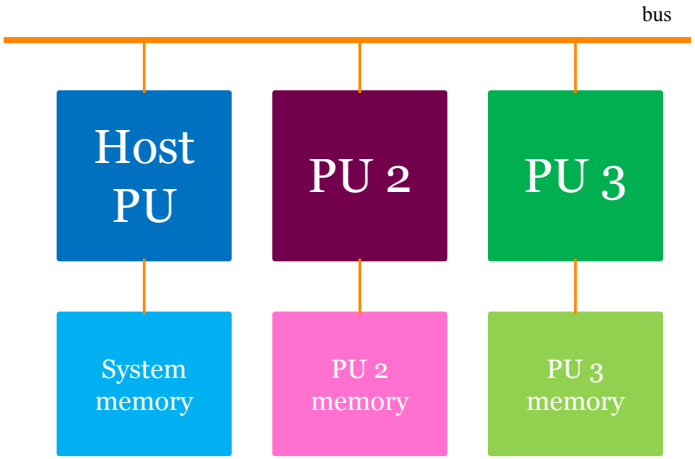
https://www.openfabrics.org/images/eventpresos/2017presentations/213_CCIXGen-Z_BBenton.pdf

https://www.computeexpresslink.org/

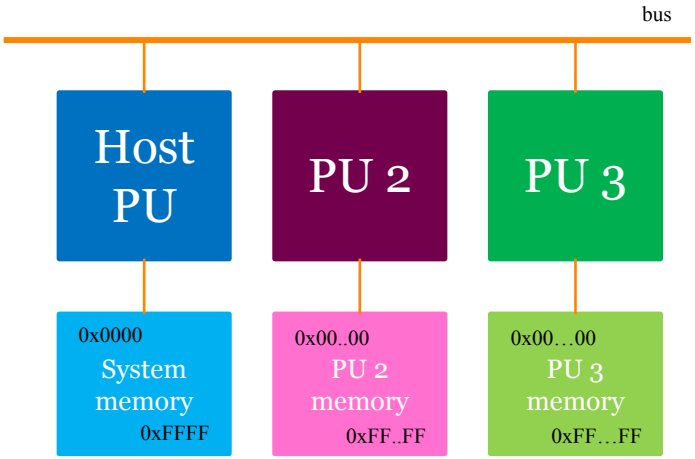
Memory models



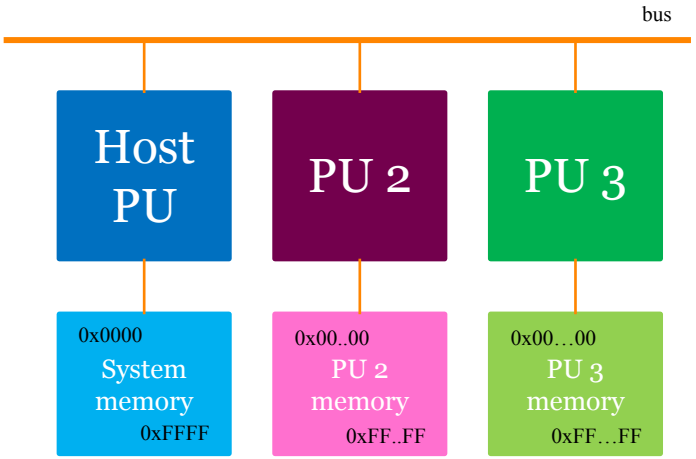
Memory models



Memory models

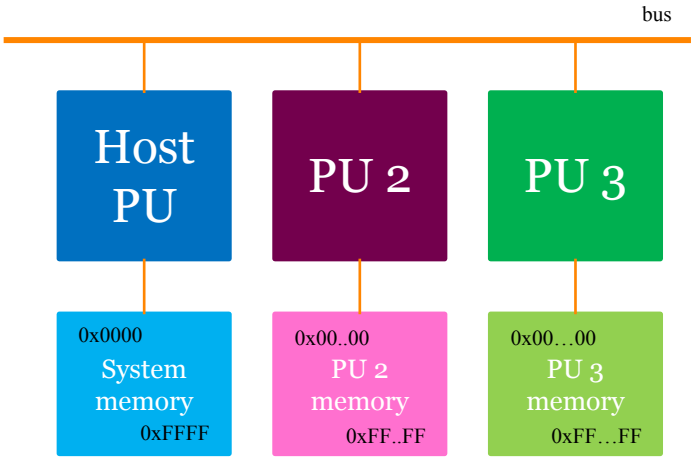


Memory models

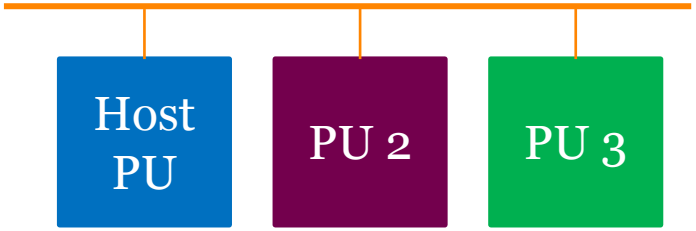


Multi Memory Model

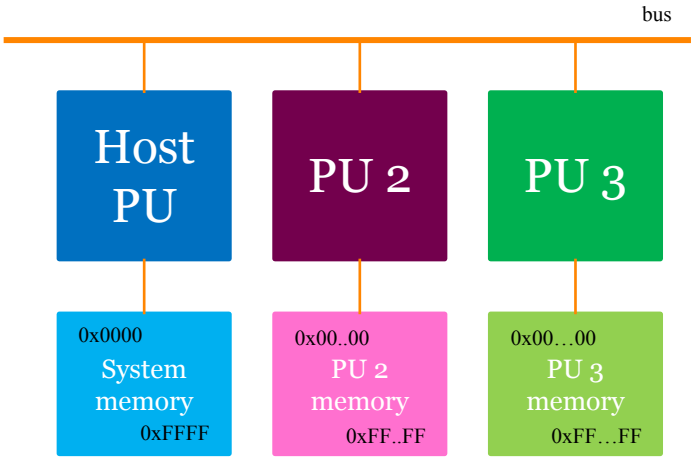
Memory models



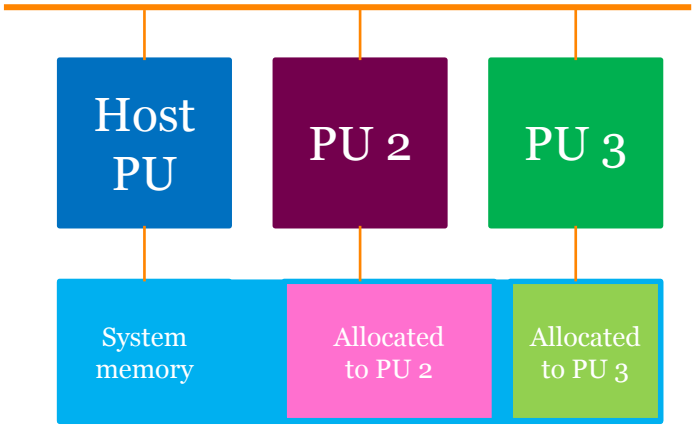
Multi Memory Model



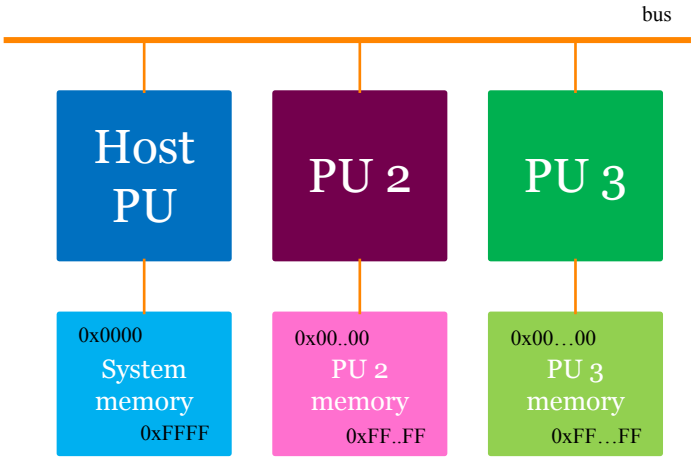
Memory models



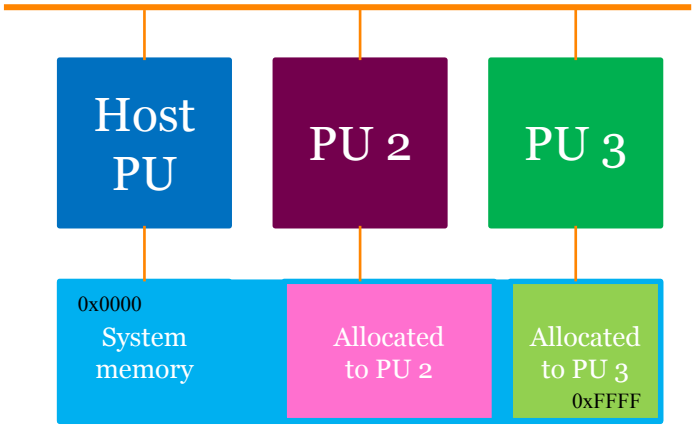
Multi Memory Model



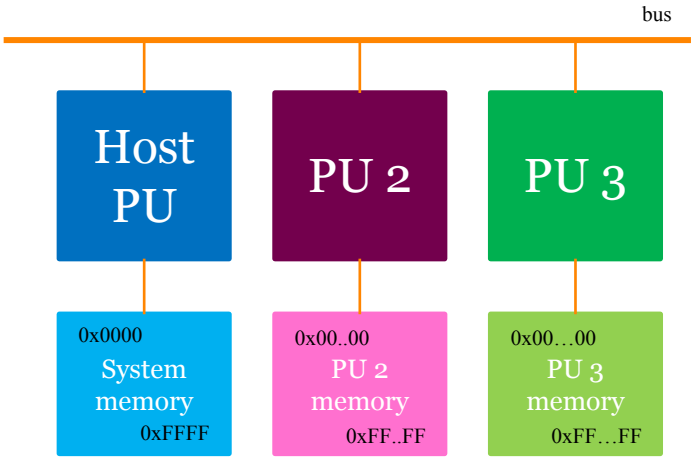
Memory models



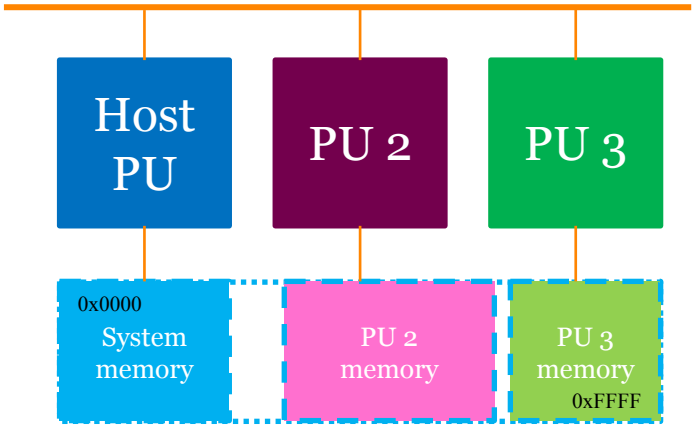
Multi Memory Model



Memory models

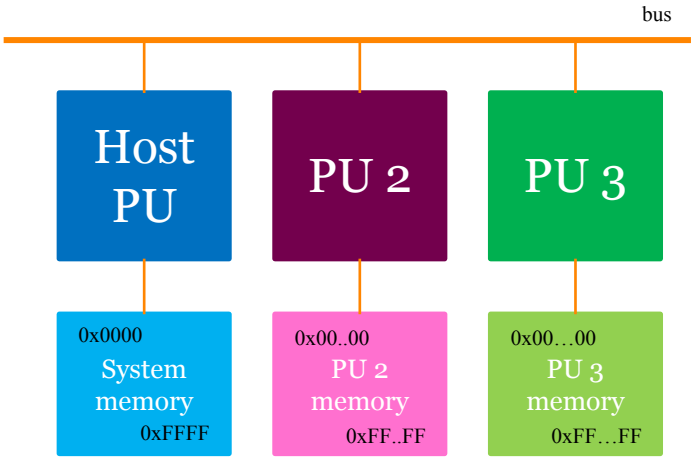


Multi Memory Model

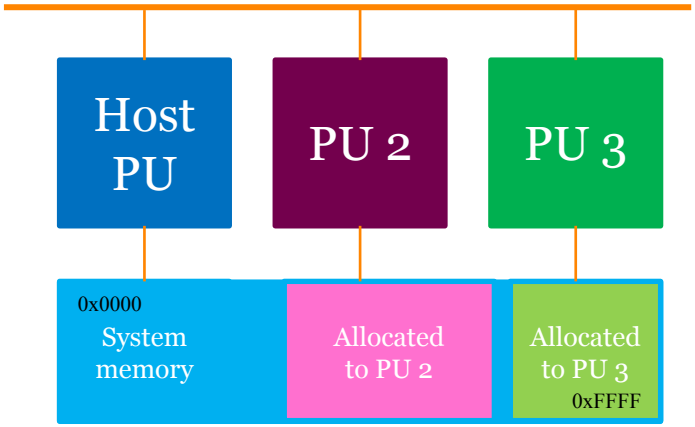


Unified (Virtual) Memory

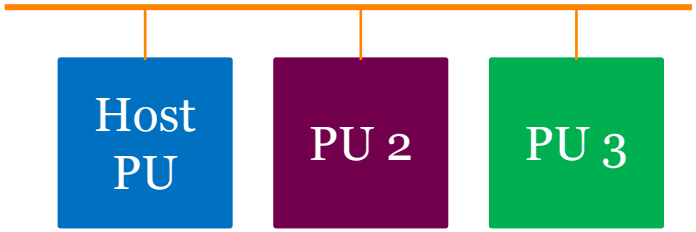
Memory models



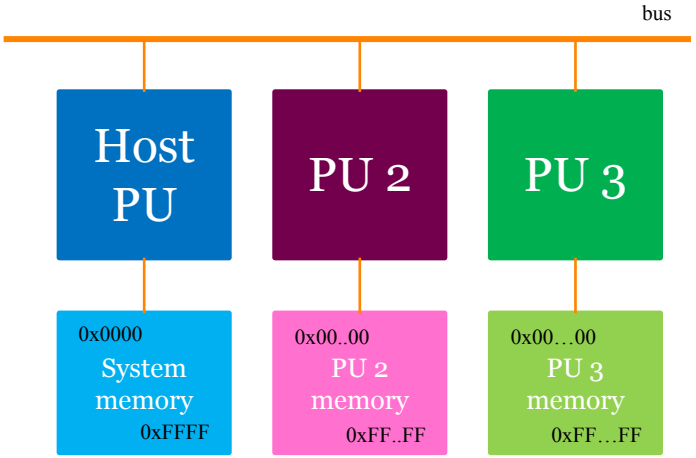
Multi Memory Model



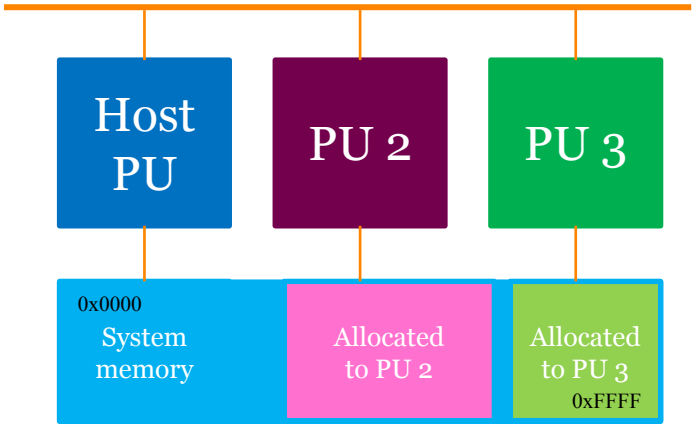
Unified (Virtual) Memory



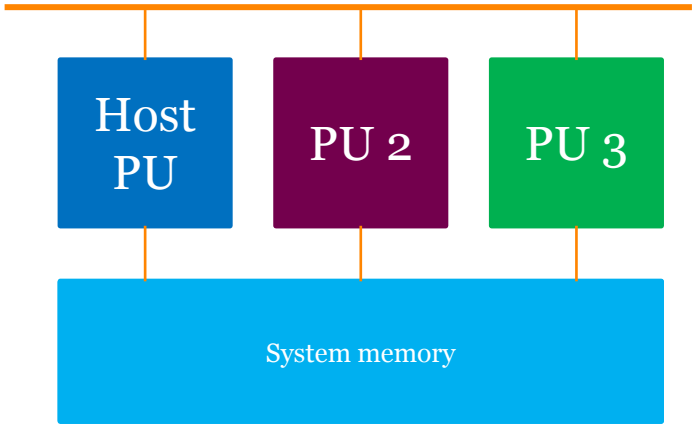
Memory models



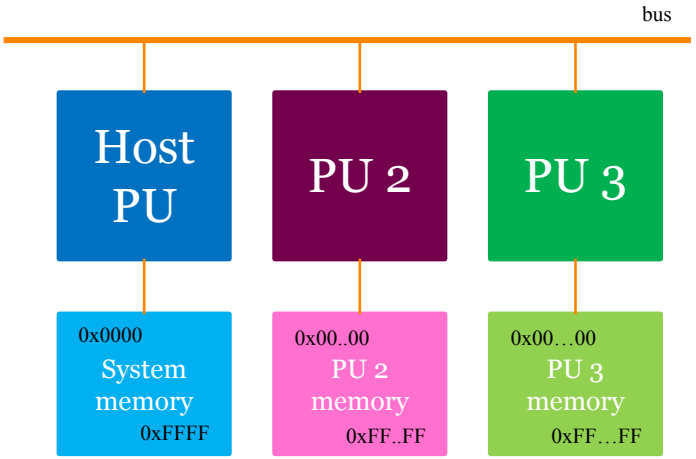
Multi Memory Model



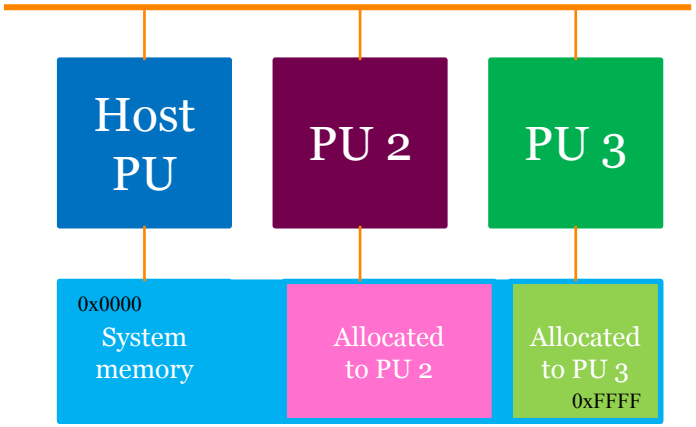
Unified (Virtual) Memory



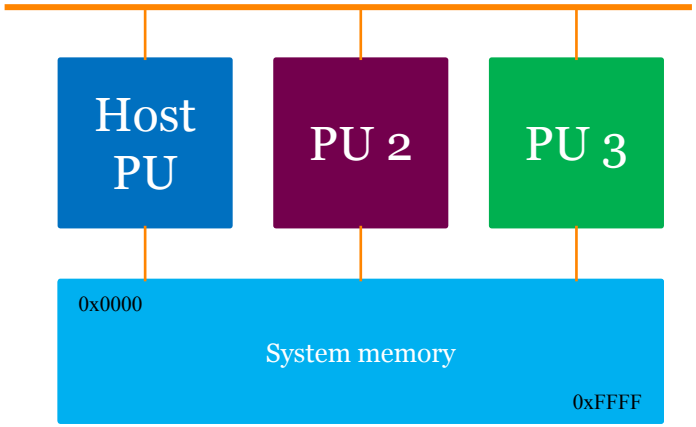
Memory models



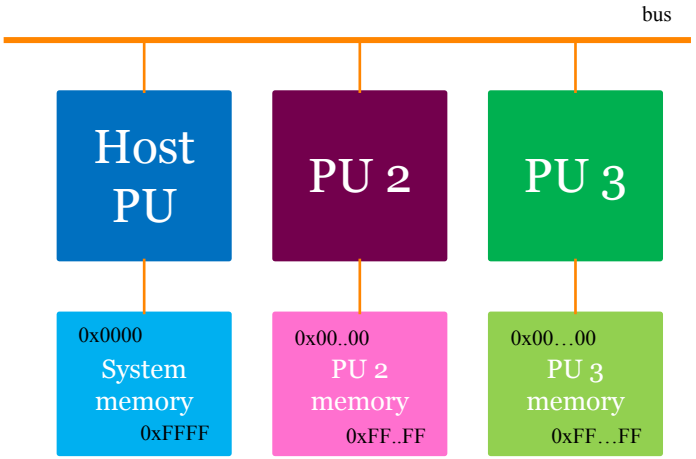
Multi Memory Model



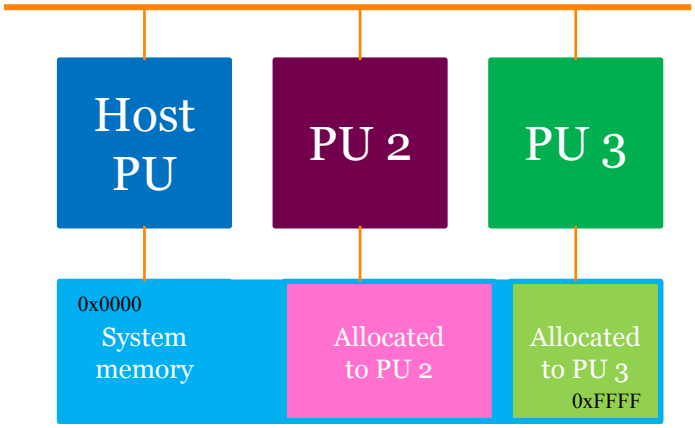
Unified (Virtual) Memory



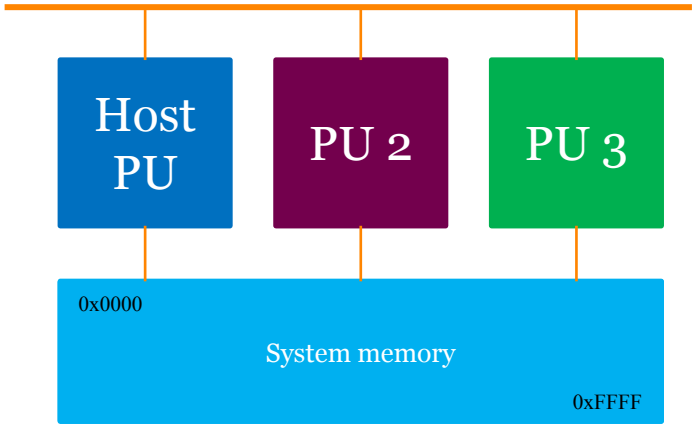
Memory models



Multi Memory Model

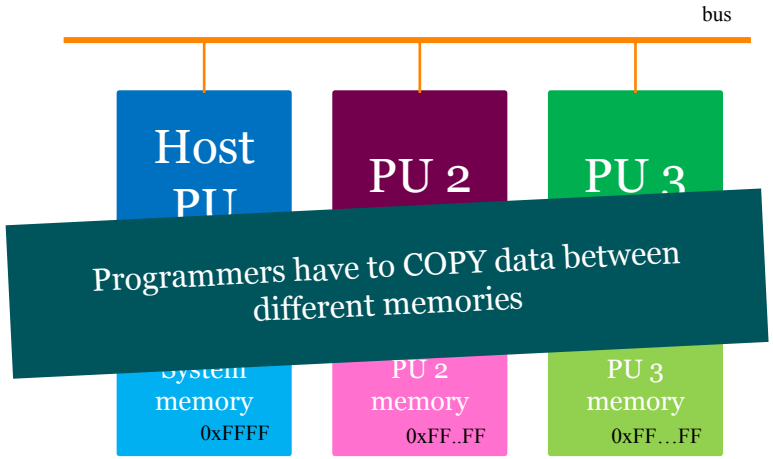


Unified (Virtual) Memory

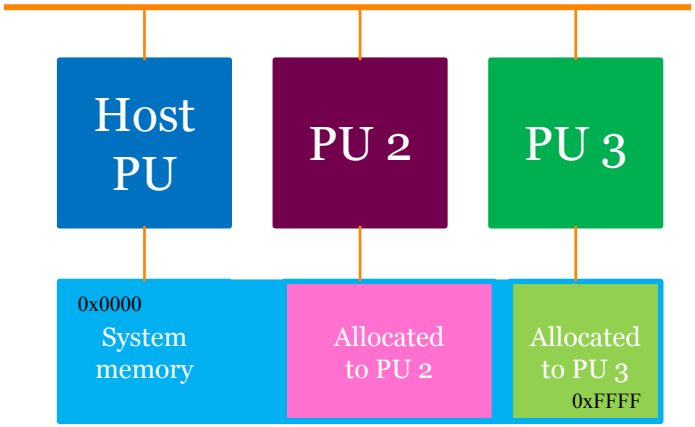


Unified Shared Memory

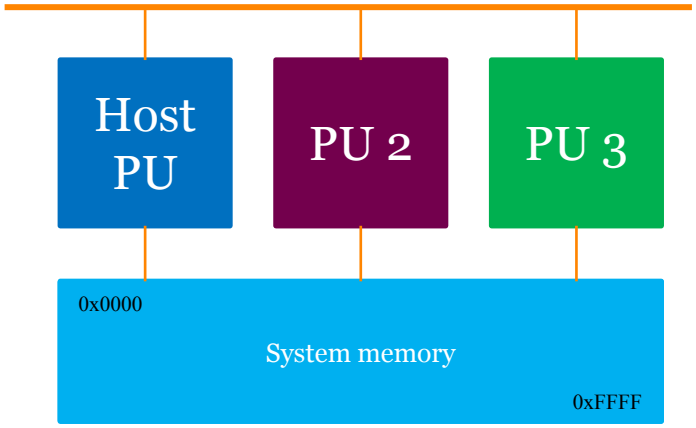
Memory models



Multi Memory Model

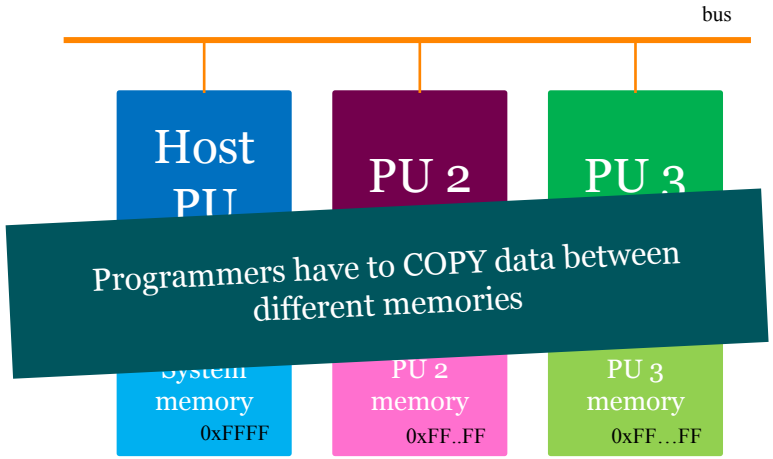


Unified (Virtual) Memory

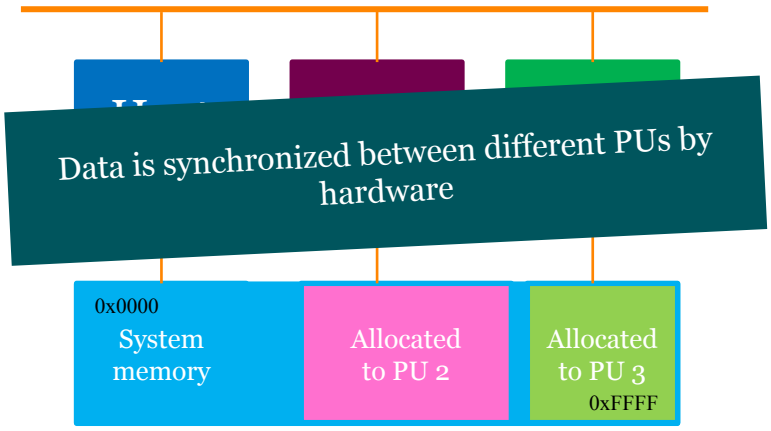


Unified Shared Memory

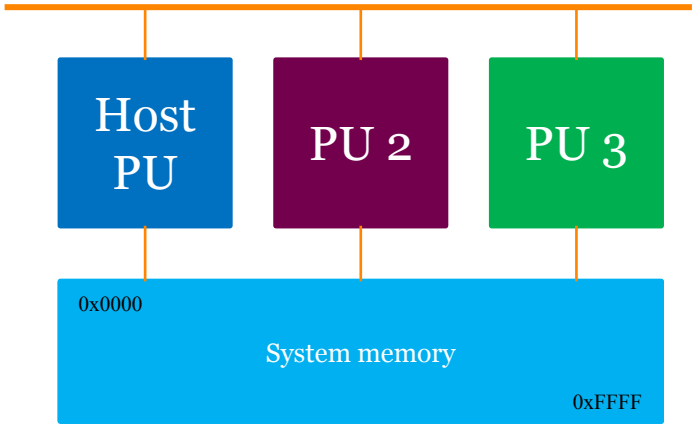
Memory models



Multi Memory Model

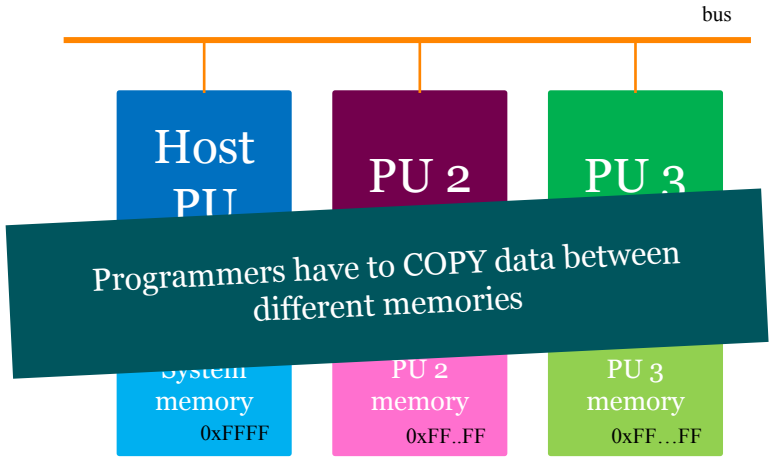


Unified (Virtual) Memory

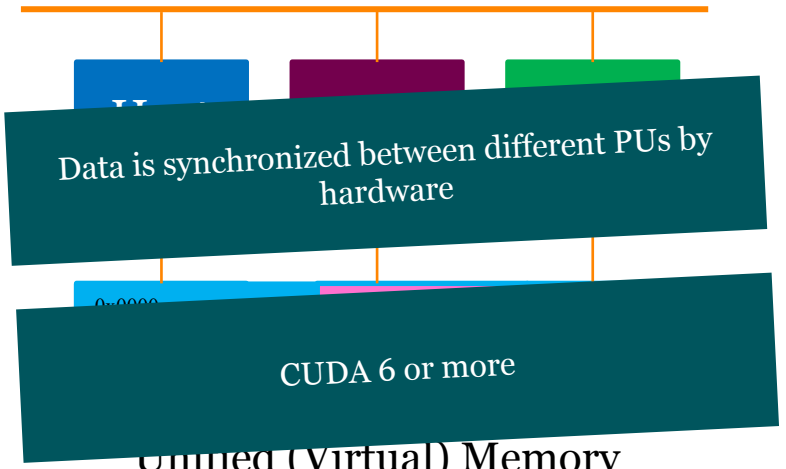


Unified Shared Memory

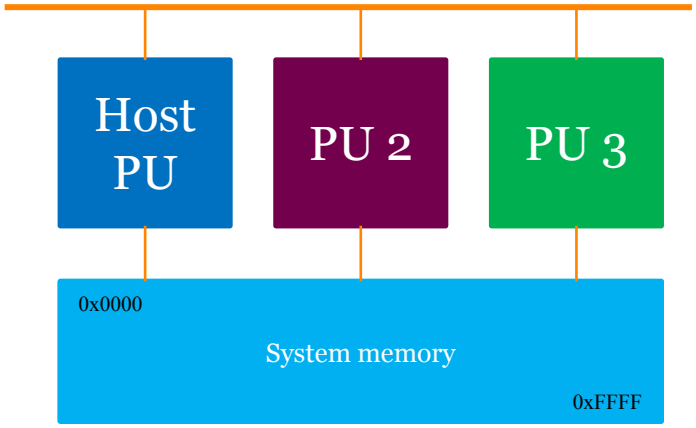
Memory models



Multi Memory Model

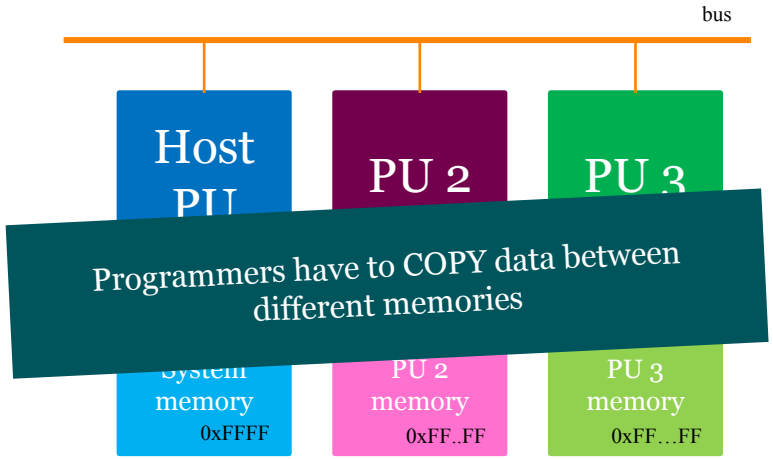


Unified (Virtual) Memory

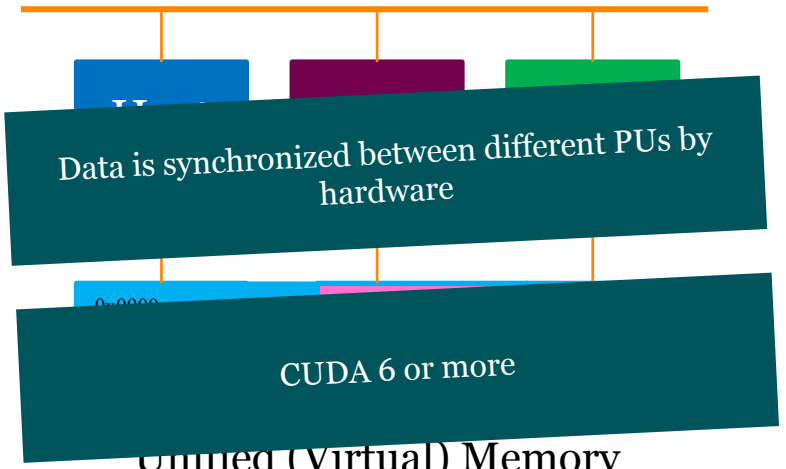


Unified Shared Memory

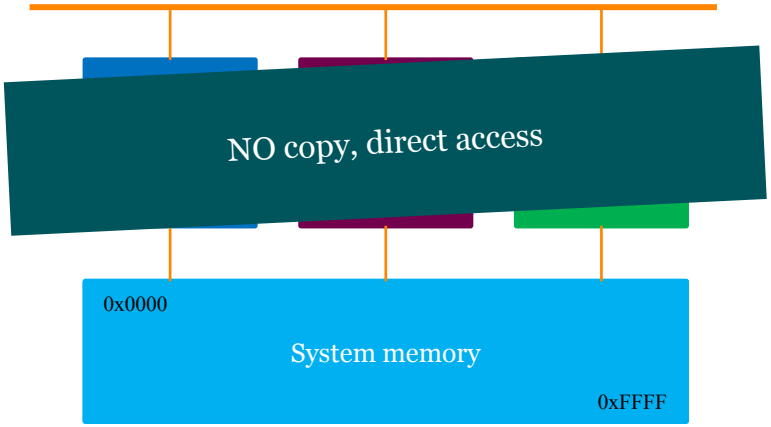
Memory models



Multi Memory Model

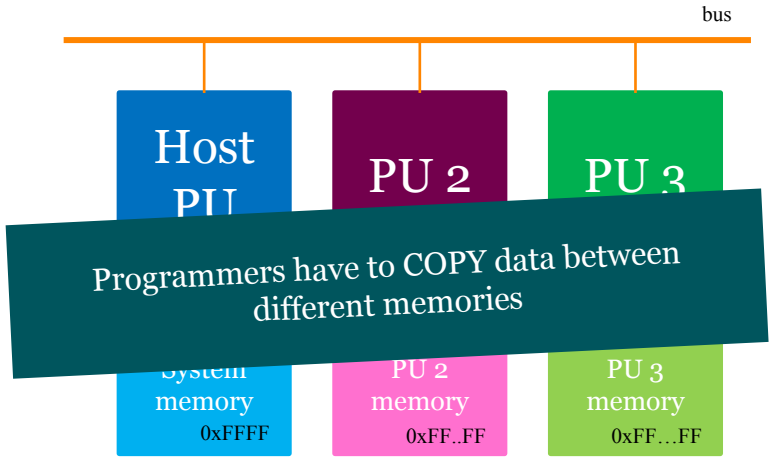


Unified (Virtual) Memory

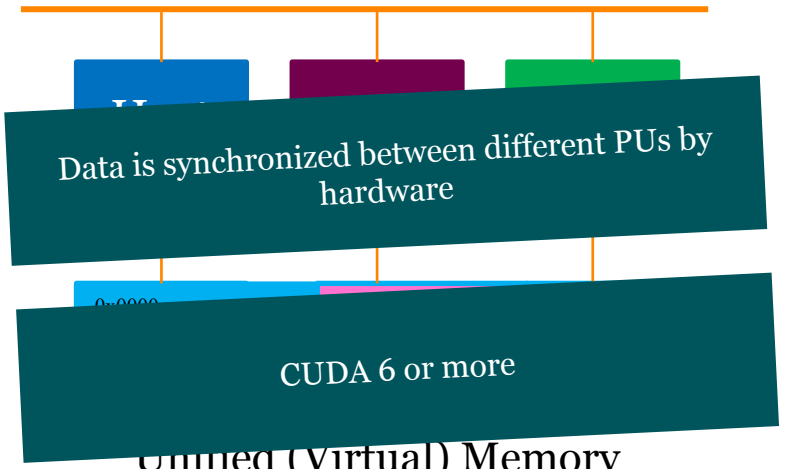


Unified Shared Memory

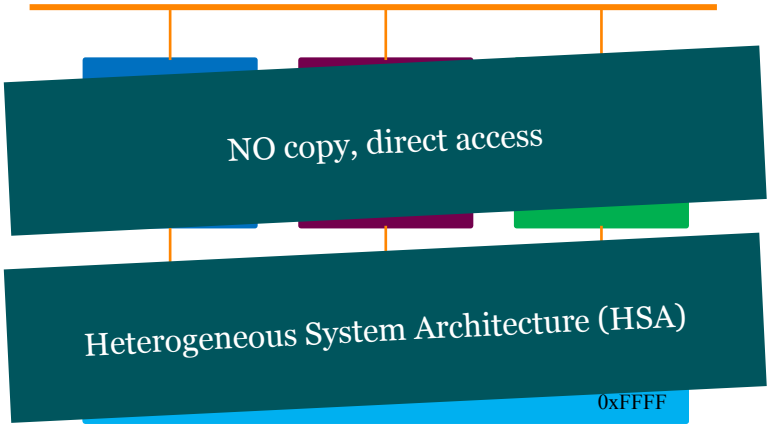
Memory models



Multi Memory Model

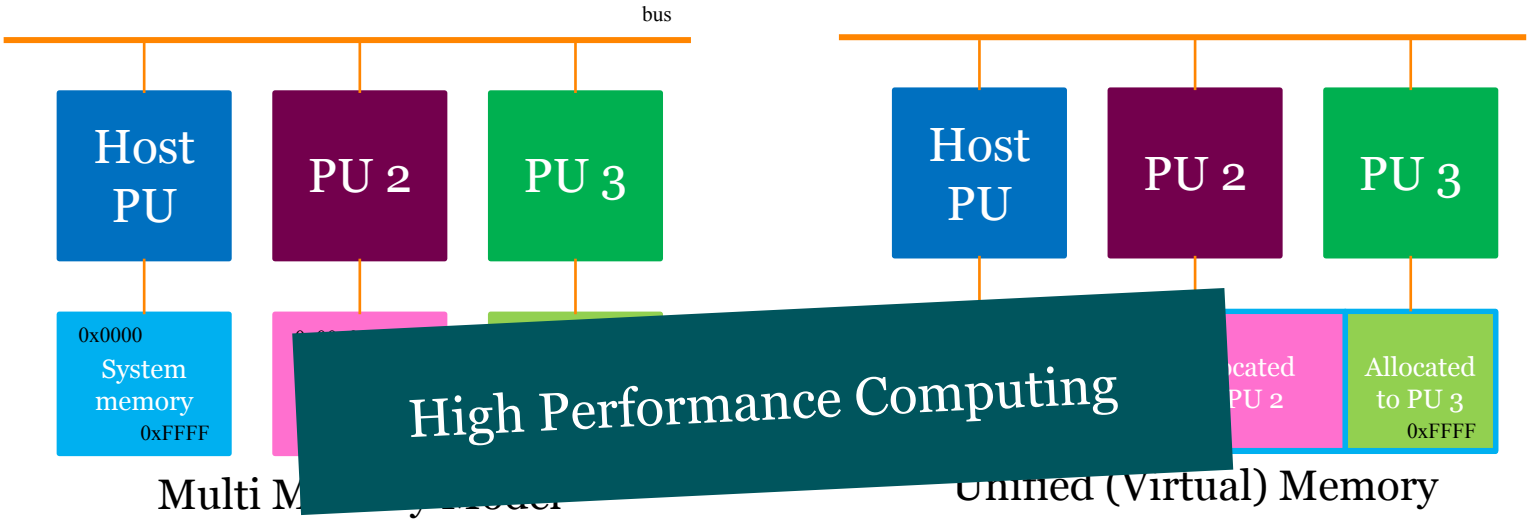


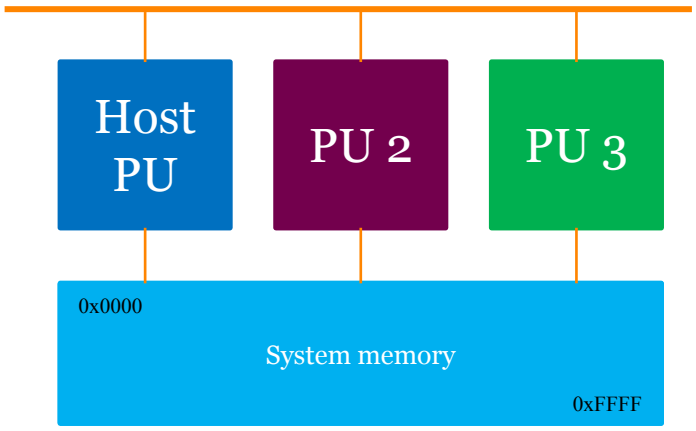
Unified (Virtual) Memory



Unified Shared Memory

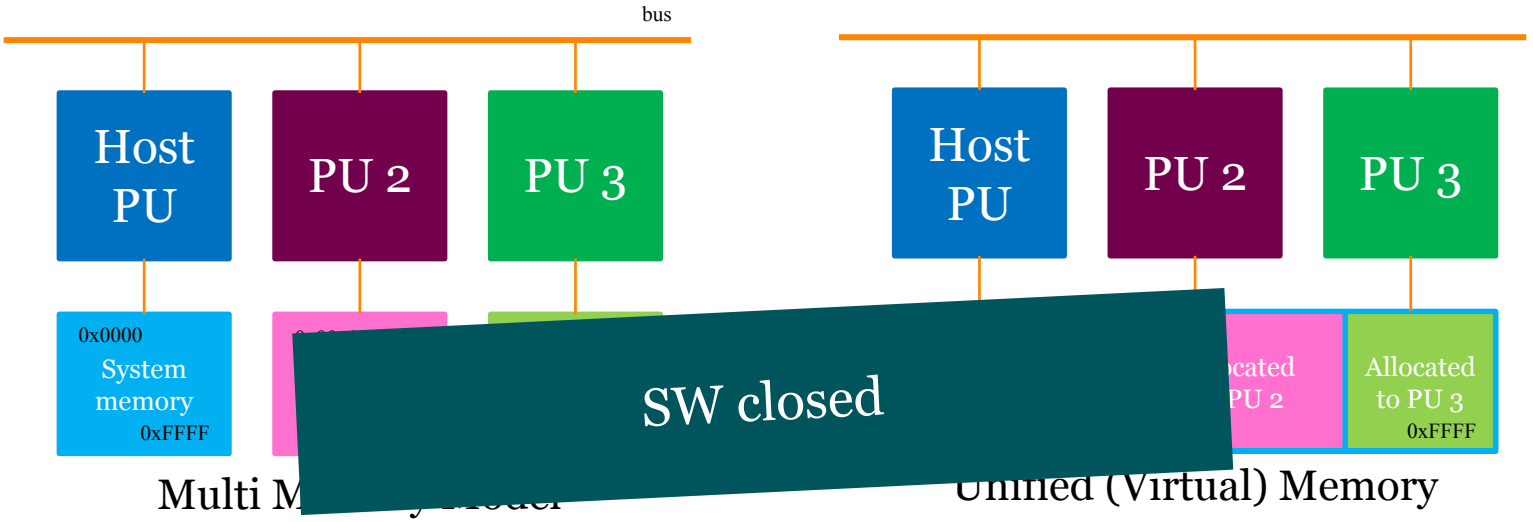
Memory models



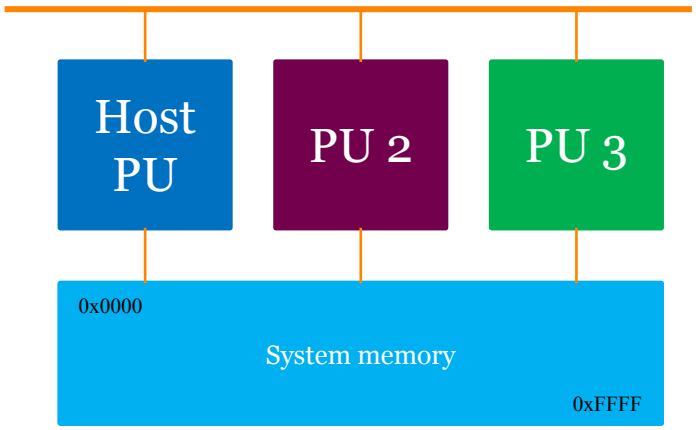


Unified Shared Memory

Memory models

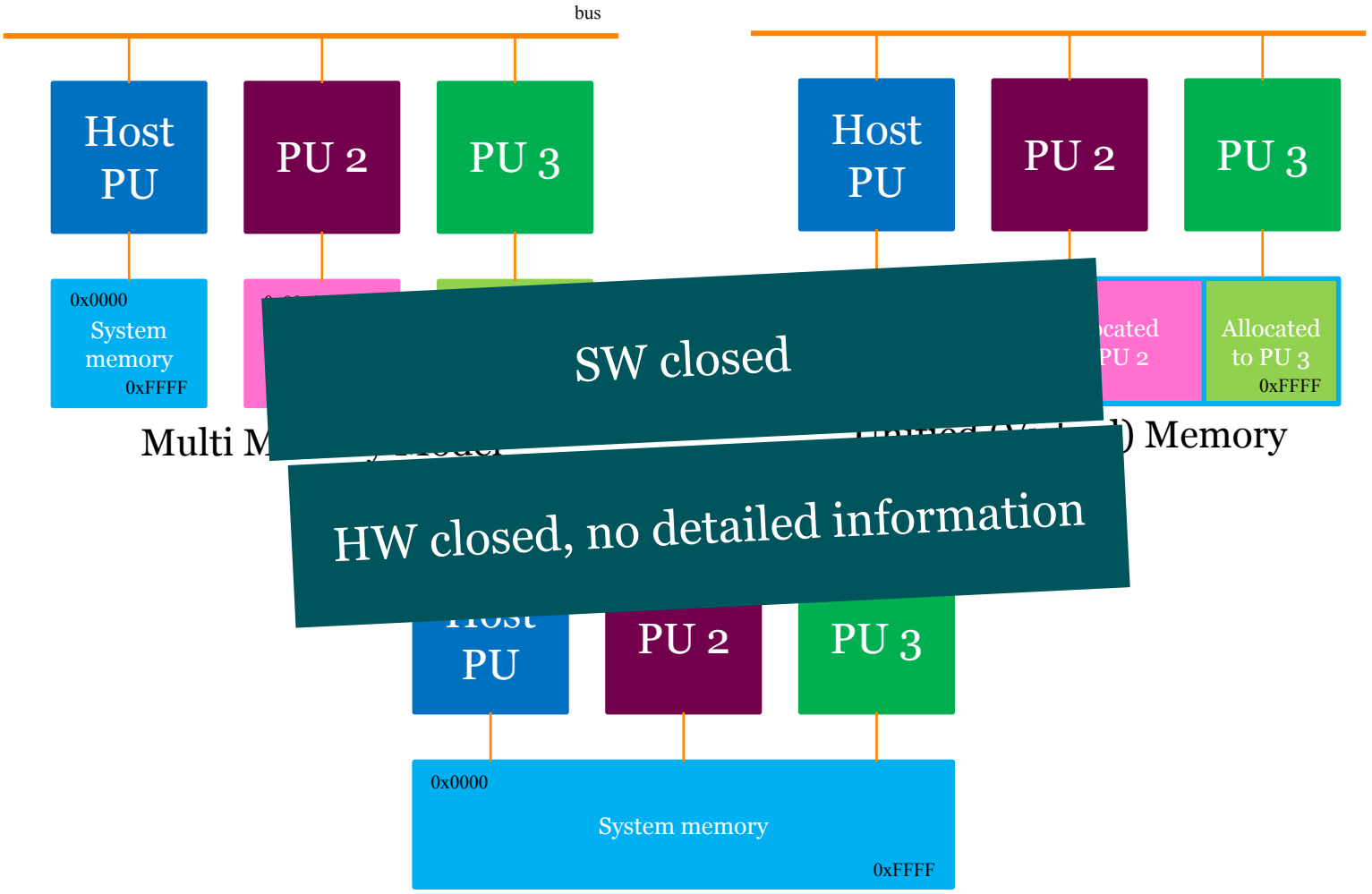


Multi M… …Unified (Virtual) Memory



Unified Shared Memory

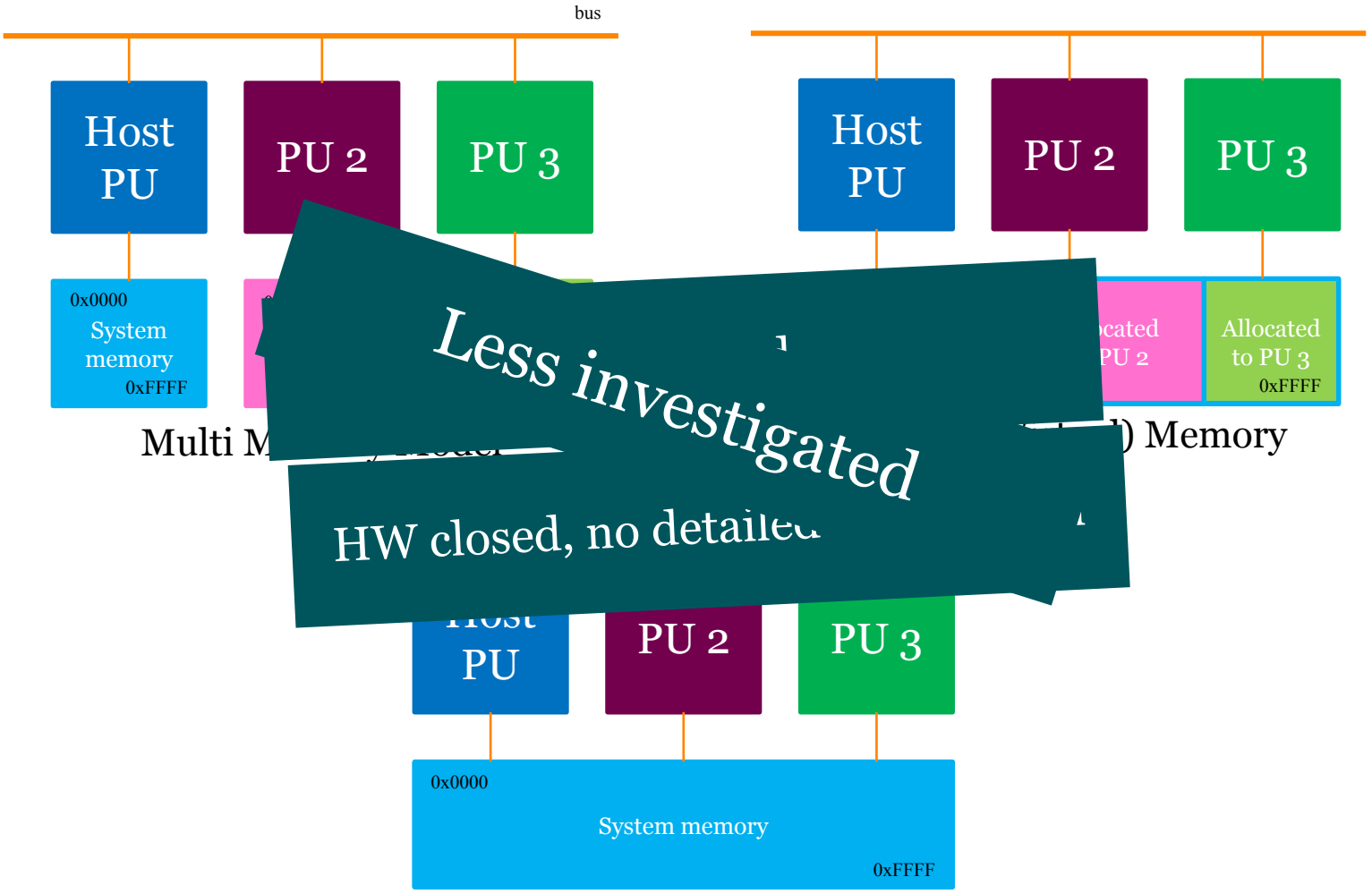
Memory models



Multi M) Memory

Unified Shared Memory

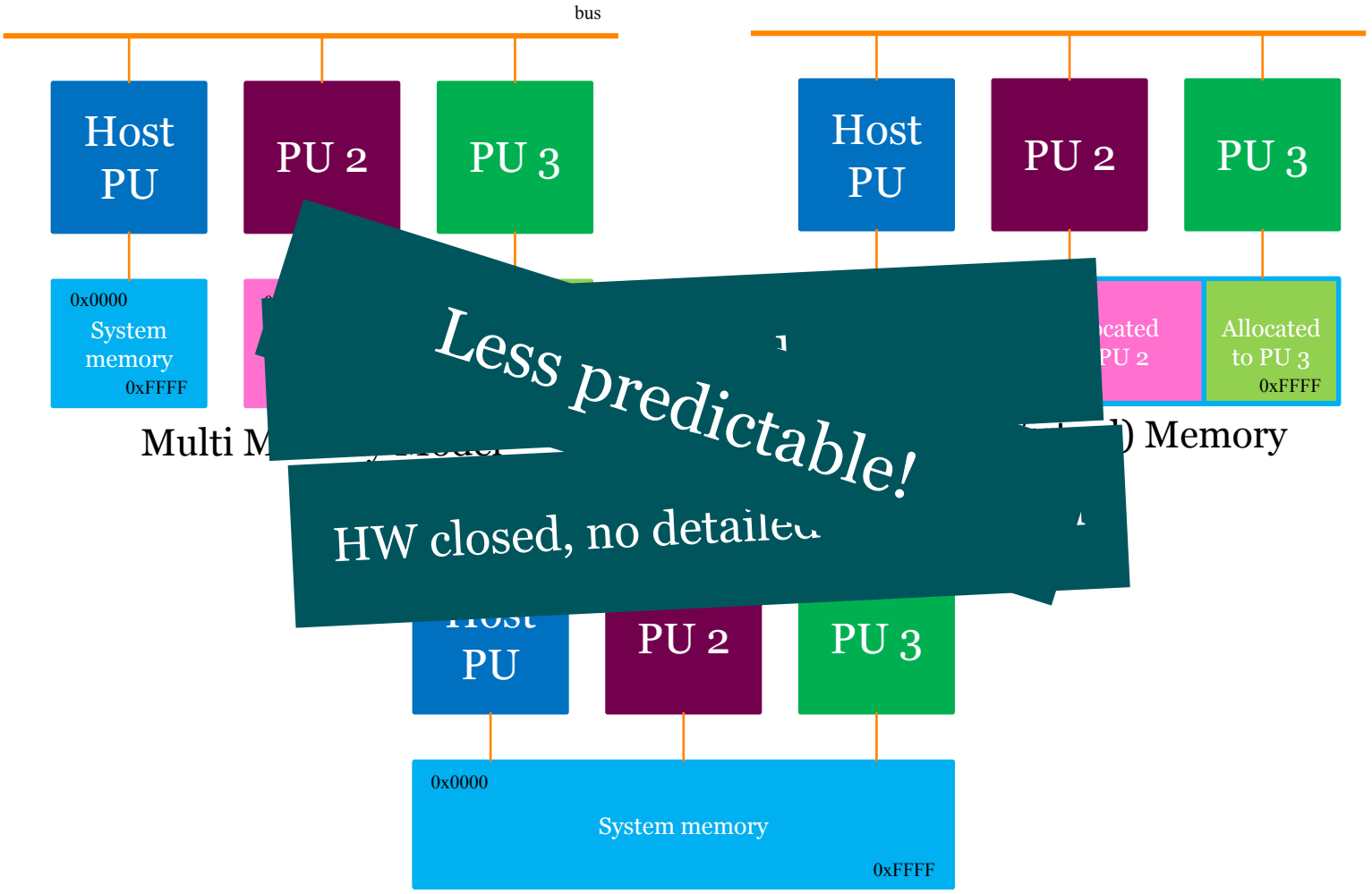
Memory models



Multi M) Memory

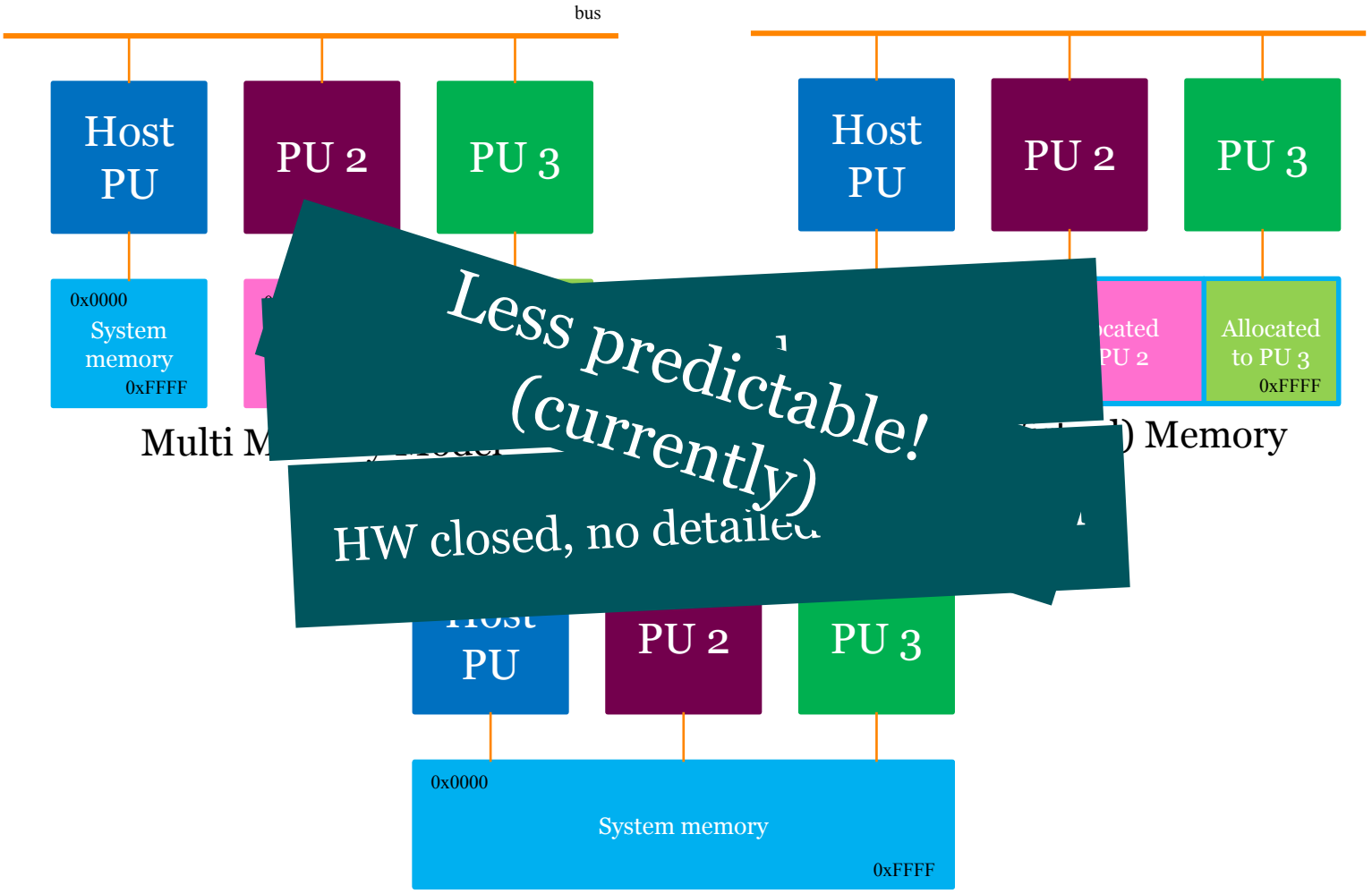
Unified Shared Memory

Memory models



Unified Shared Memory

Memory models



Memory models
Standards & Specifications

- High Bandwidth Memory (HBM)
- Embedded Multi-Die Interconnect Bridge (EMIB)
- Hybrid Memory Cube (HMC)
- 3D XPoint
- HBM2

Heterogeneous computing Misc

- Languages /Frameworks
  - OpenCL
  - OpenMP
  - CUDA
  - HIP
  - C++ AMP etc.

Heterogeneous computing Misc

- Languages /Frameworks
  - OpenCL
  - OpenMP
  - CUDA
  - HIP
  - C++ AMP etc.
- Method
  - Pinned Memory
  - Pipeline
  - Asynchronous Transfers
  - Persistent kernel/thread

Heterogeneous computing Misc

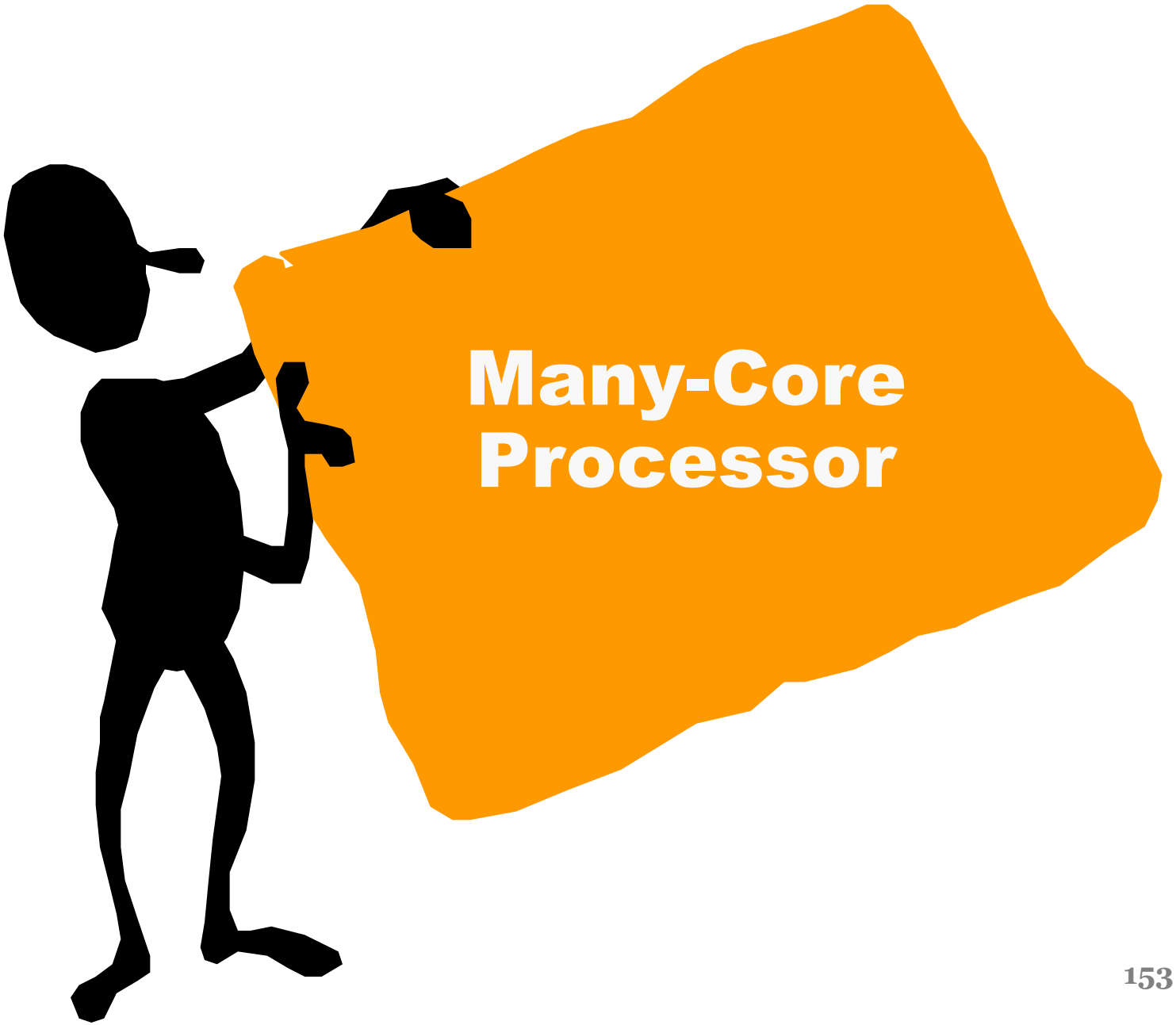
- Languages /Frameworks
  - OpenCL
  - CUDA
  - HIP etc.
- Method
  - Pinned Memory
  - Pipeline
  - Asynchronous Transfers
  - Persistent kernel/thread
- Specifications
  - HSA (Heterogeneous System Architecture)
  - CCIX, Gen-Z, OpenCAPI
  - HBM, EMIB, HMC, 3D XPoint, HBM2

Heterogeneous computing Misc

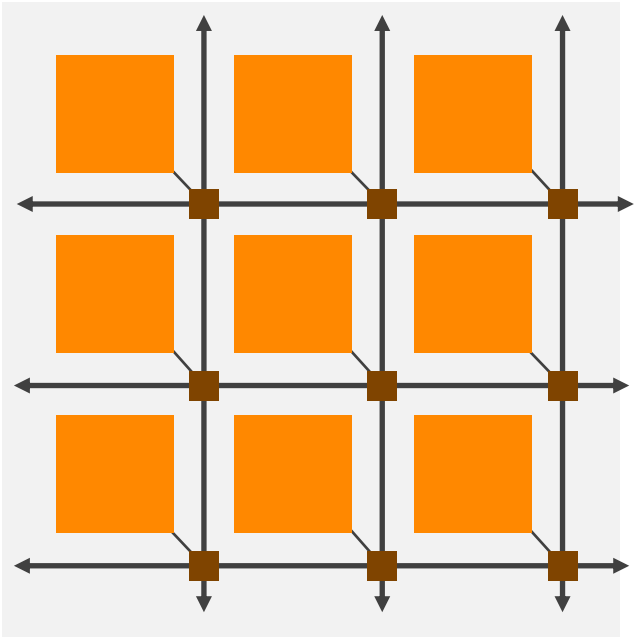
- Languages /Frameworks
  - OpenCL
  - CUDA
  - HIP etc.
- COTS Platforms
  - AMD APUs
    - AMD CPU + AMD GPU
    - Carrizo, Ryzen and so on
  - NVIDIA Jetson X1/X2
    - ARM CPU + NVIDIA GPU
  - Xilinx Ultrascale
    - ARM CPU + ARM GPU + Xilinx FPGA
  - Intel Stratix 10 FPGA SoC
    - ARM CPU + Intel(Altera) FPGA
- Method
  - Pinned Memory
  - Pipeline
  - Asynchronous Transfers
  - Persistent kernel/thread
- Specifications
  - HSA (Heterogeneous System Architecture)
  - CCIX, Gen-Z, OpenCAPI
  - HBM, EMIB, HMC, 3D XPoint, HBM2

Heterogeneous computing Misc

- Languages /Frameworks
  - OpenCL
  - CUDA
  - HIP etc.
- COTS Platforms
  - AMD APUs
    - AMD CPU + AMD GPU
    - Carrizo, Ryzen and so on
  - NVIDIA Jetson X1/X2
    - ARM CPU + NVIDIA GPU
  - Xilinx Ultrascale
    - ARM CPU + ARM GPU + Xilinx FPGA
  - Intel Stratix 10 FPGA SoC
    - ARM CPU + Intel(Altera) FPGA
- Method
  - Pinned Memory
  - Pipeline
  - Asynchronous Transfers
  - Persistent kernel/thread
- Specifications
  - HSA (Heterogeneous System Architecture)
  - CCIX, Gen-Z, OpenCAPI
  - HBM, EMIB, HMC, 3D XPoint, HBM2
- Papers
  - Sparsh Mittal and Jeffrey Vetter, "*A survey of CPU-GPU heterogeneous computing techniques*"
  - Hyoseung Kim et al., *"A Server-based Approach for Predictable GPU Access Control"*


Many-Core
Processor

Many-Core Processor



- Main elements:
  - Tiles that host the compute cores
  - Network to connect the tiles

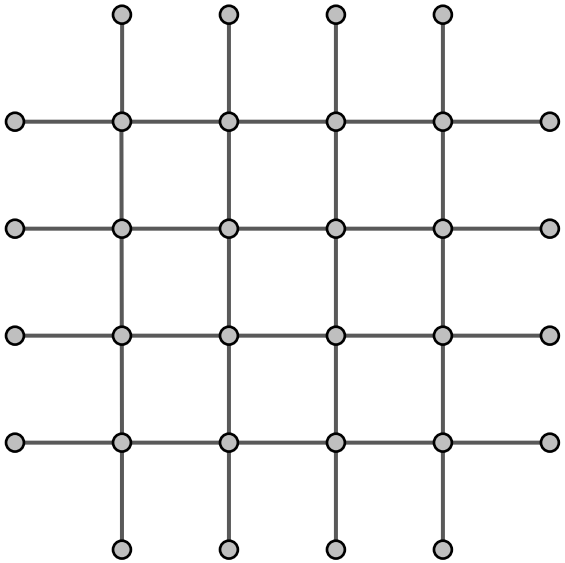
Interconnect

Network-on-Chip

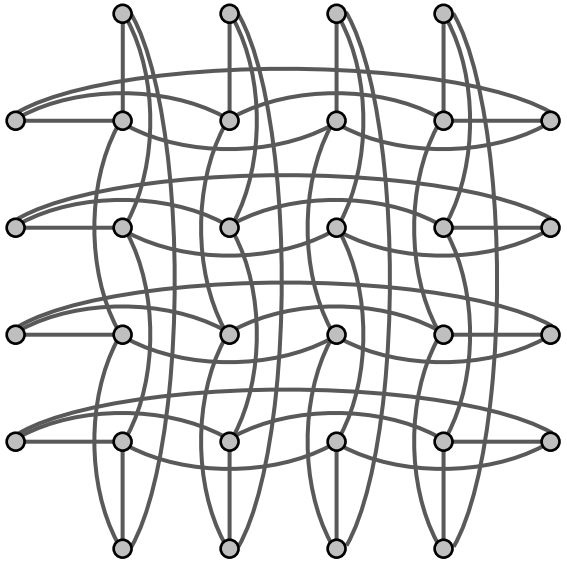
- The many-core hosts a larger amount of elements than multi-cores
  - Traditional interconnects do not scale
  - → Network-on-Chip is used as interconnect

Interconnect
Network-on-Chip

- The many-core hosts a larger amount of elements than multi-cores
  - Traditional interconnects do not scale
  - → Network-on-Chip is used as interconnect
- Different topologies possible
  - For example:



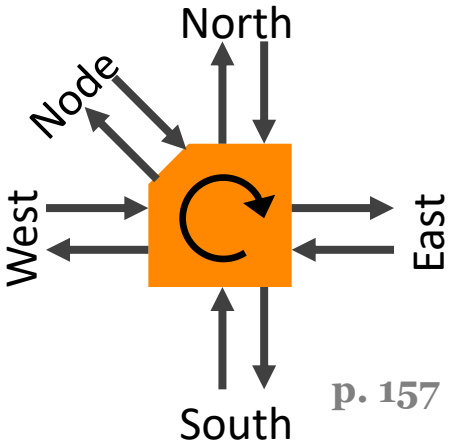
2D-Mesh



2D-Torus

Interconnect
Network-on-Chip

- NoC elements
  - Links that connect NoC elements to each other
    - Routers
    - Source and sink nodes
  - Source and sink nodes inject and receive messages
  - Routers forward the messages on the network
    - Buffers
    - Arbitration policy



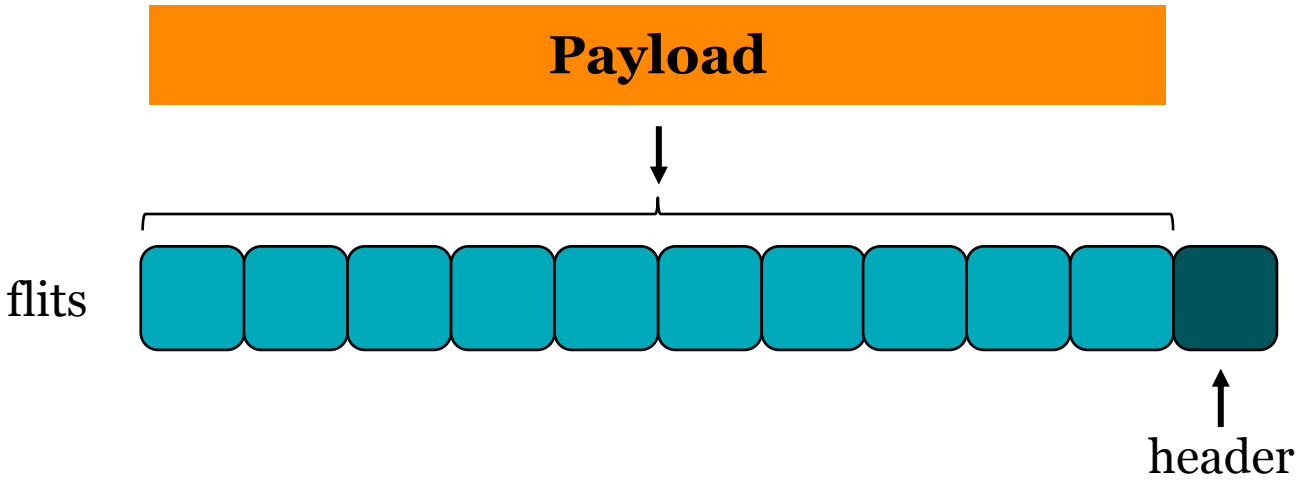
Interconnect

Network-on-Chip

- NoC applies wormhole switching
  - Messages do not need to be stored completely at intermediate routers
    - Low buffer requirement compared to store and forward switching (ethernet)

Interconnect
Network-on-Chip

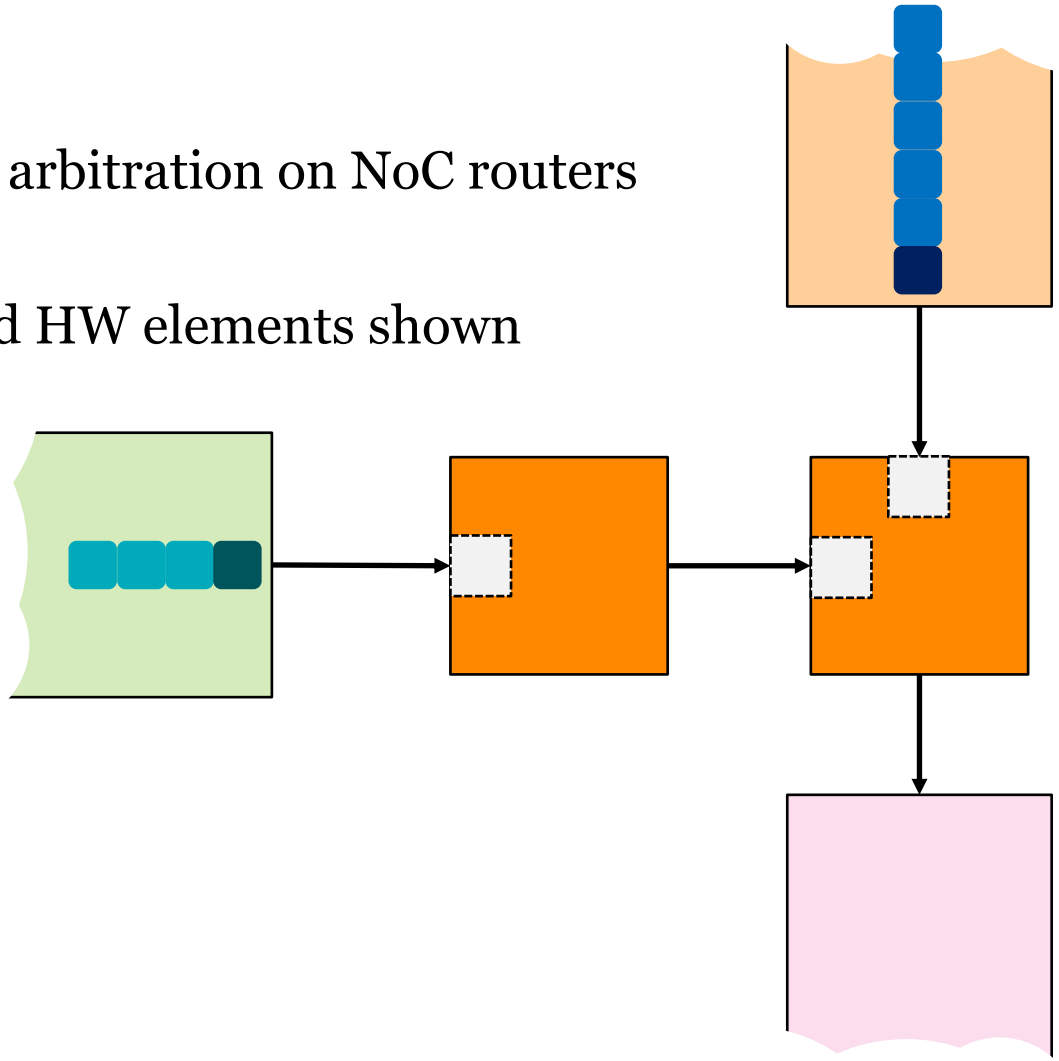
- NoC applies wormhole switching
  - Messages do not need to be stored completely at intermediate routers
    - Low buffer requirement compared to store and forward switching (ethernet)
  - A message is split into **flits** (flow control digits)
  - A header flit is appended to store routing information



Wormhole Switching

Example

- Round robin arbitration on NoC routers
- 2 messages
- Only involved HW elements shown

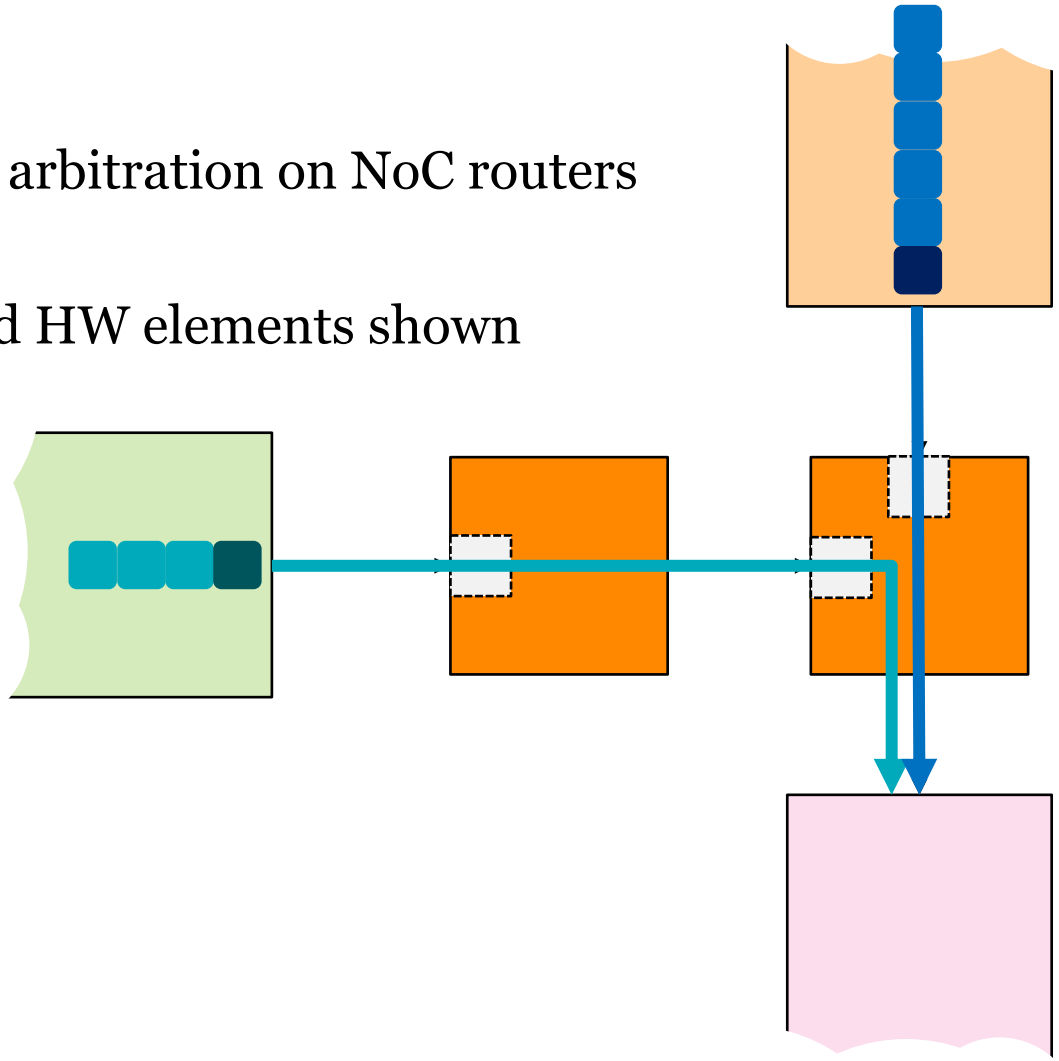


Counter: 0

Wormhole Switching

Example

- Round robin arbitration on NoC routers
- 2 messages
- Only involved HW elements shown

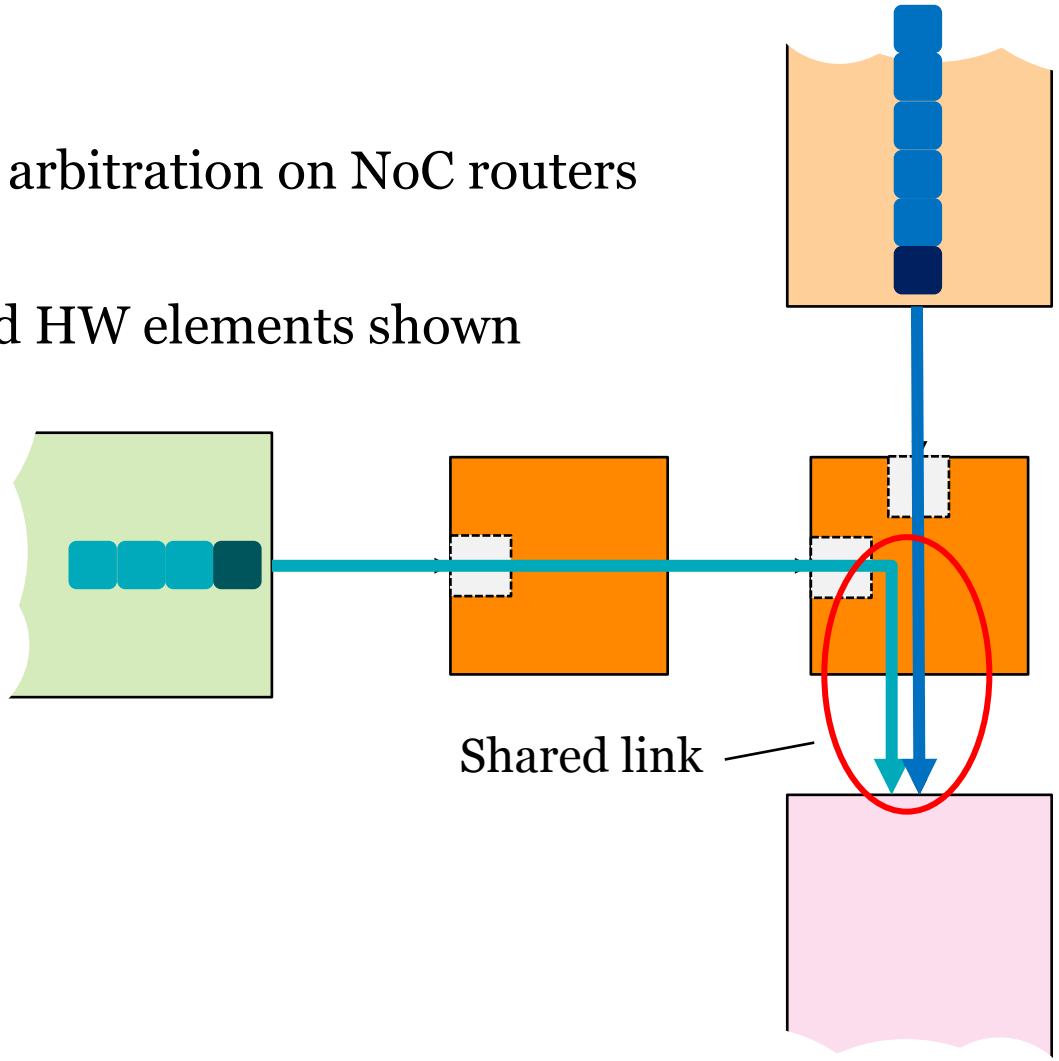


Counter: 0

Wormhole Switching

Example

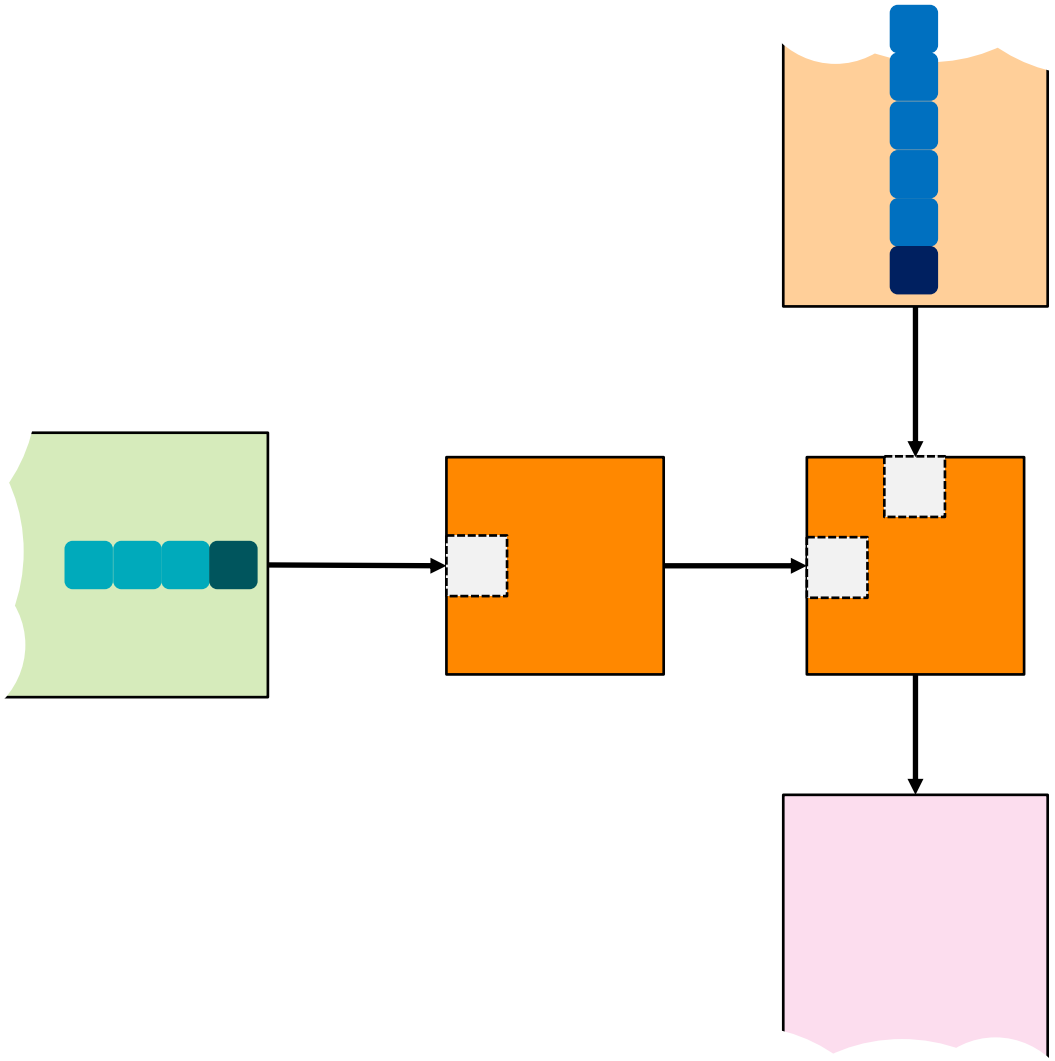
- Round robin arbitration on NoC routers
- 2 messages
- Only involved HW elements shown



Counter: 0

Wormhole Switching

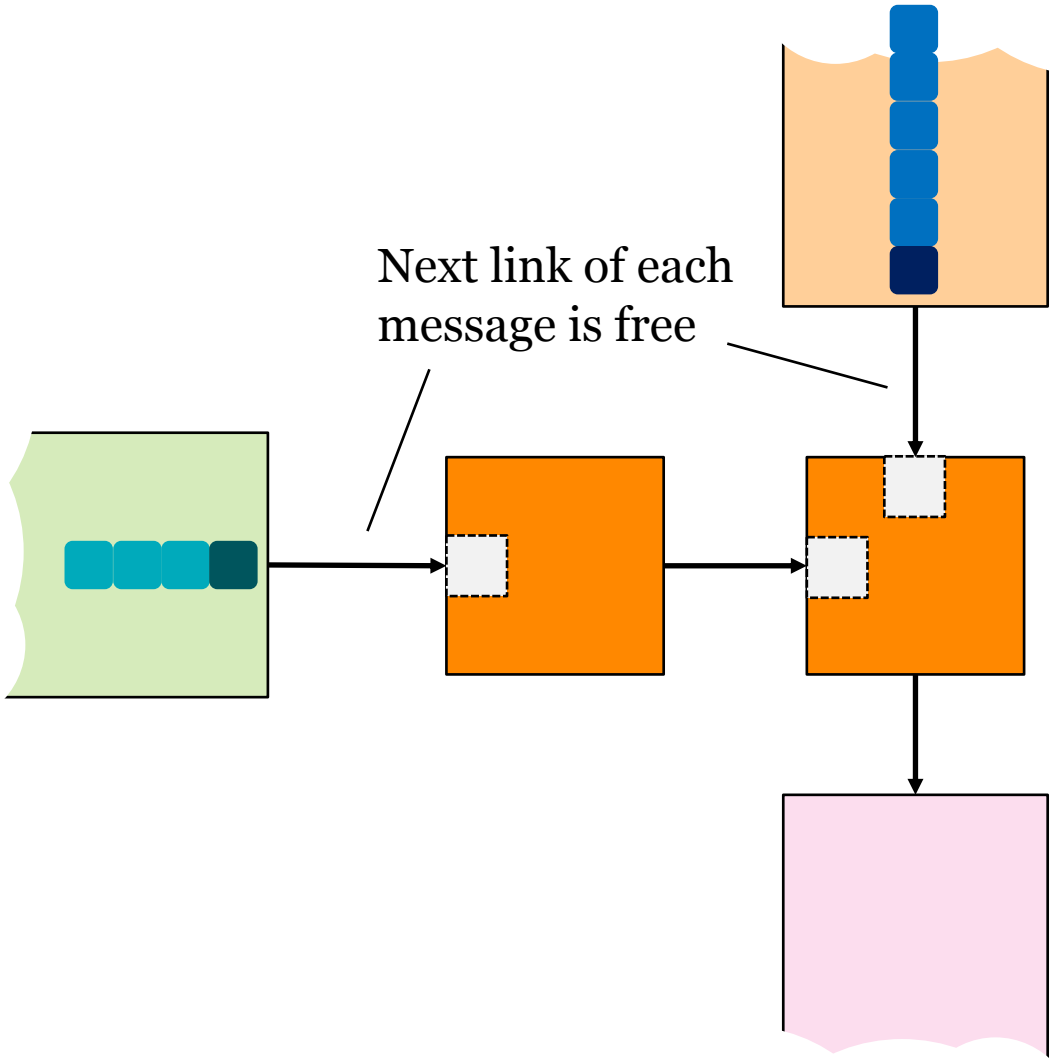
Example



Counter: 0

Wormhole Switching

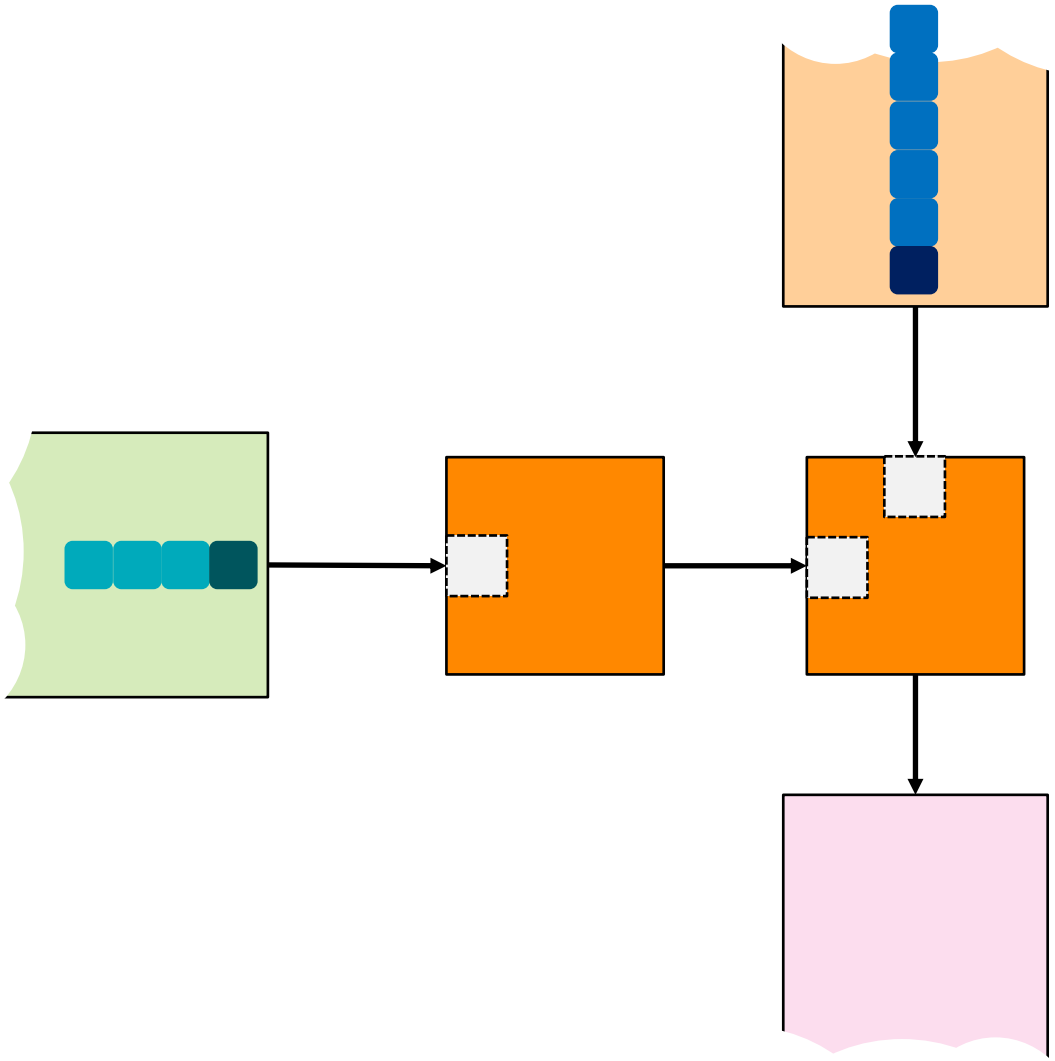
Example



Counter: 0

Wormhole Switching

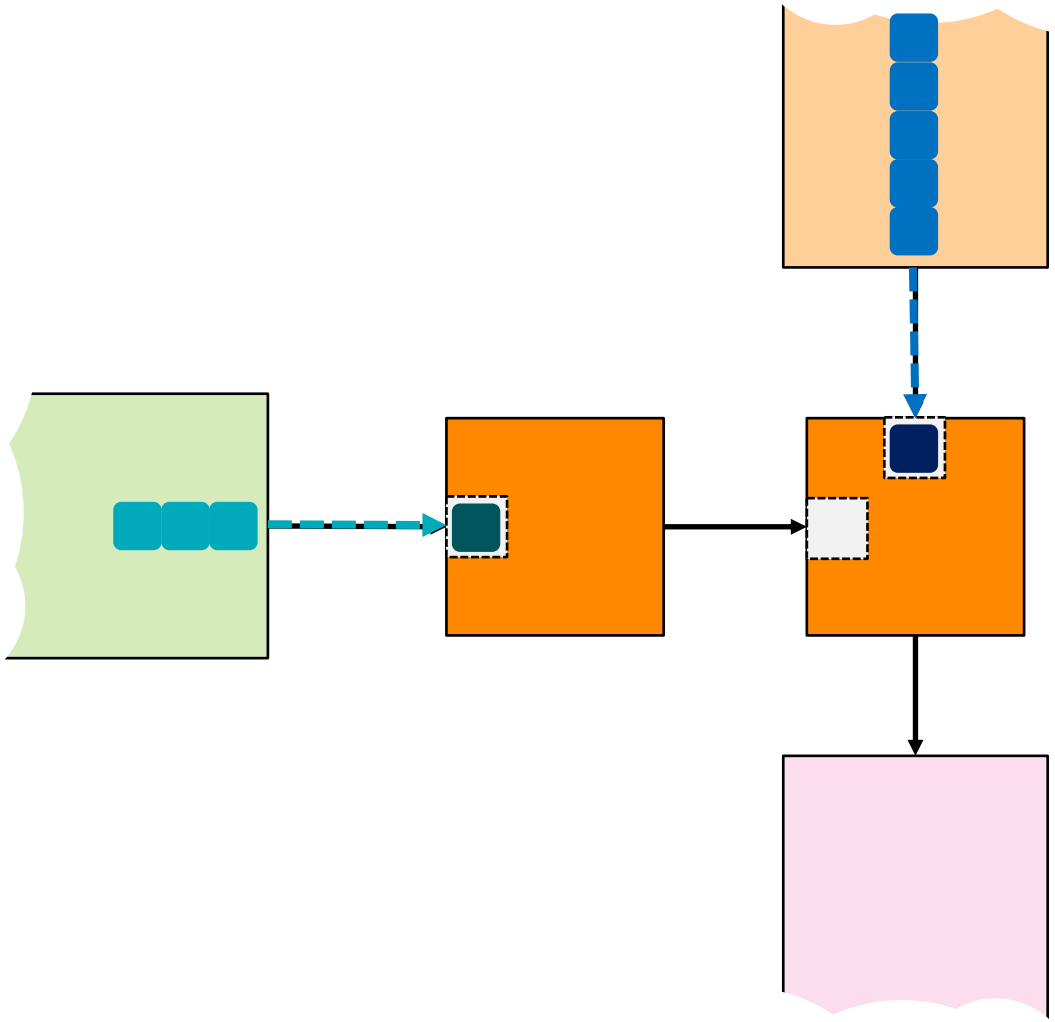
Example



Counter: 0

Wormhole Switching

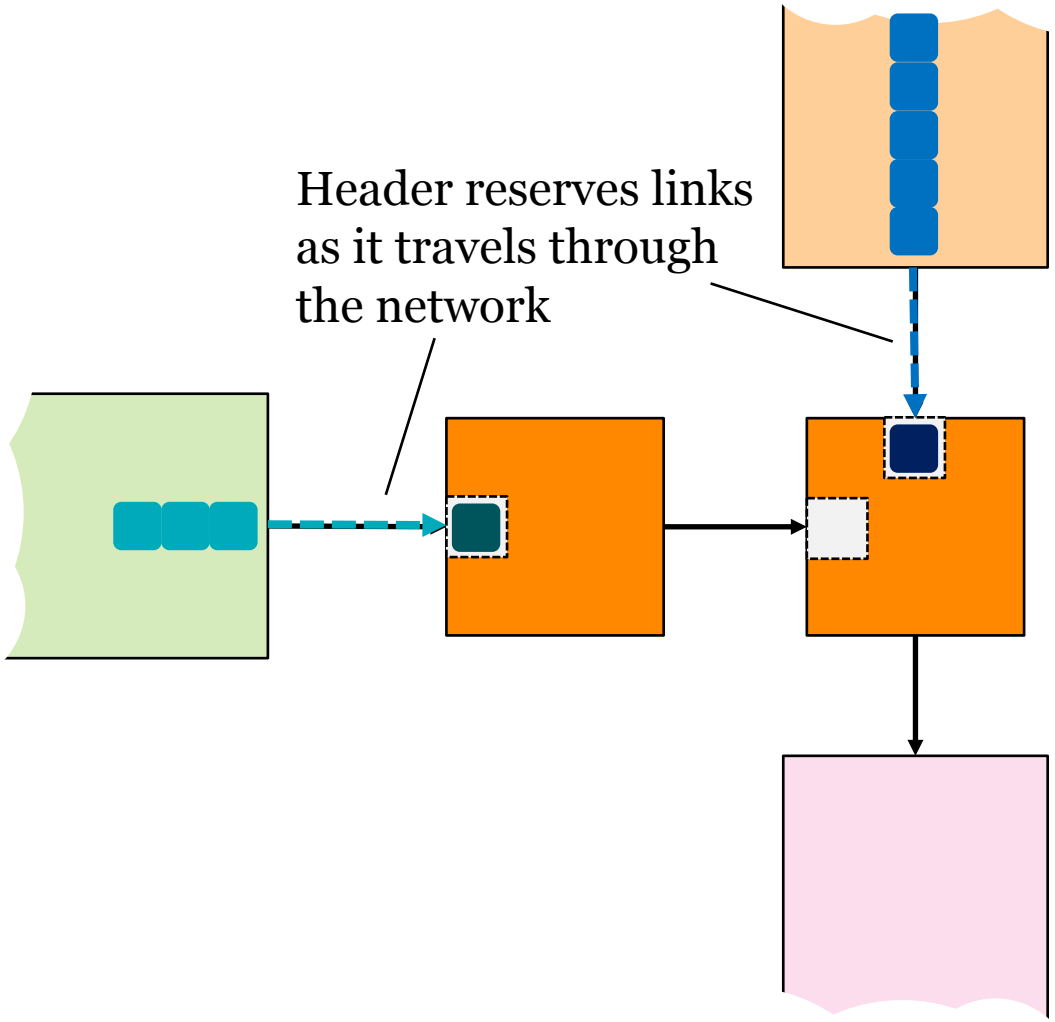
Example



Counter: 1

Wormhole Switching

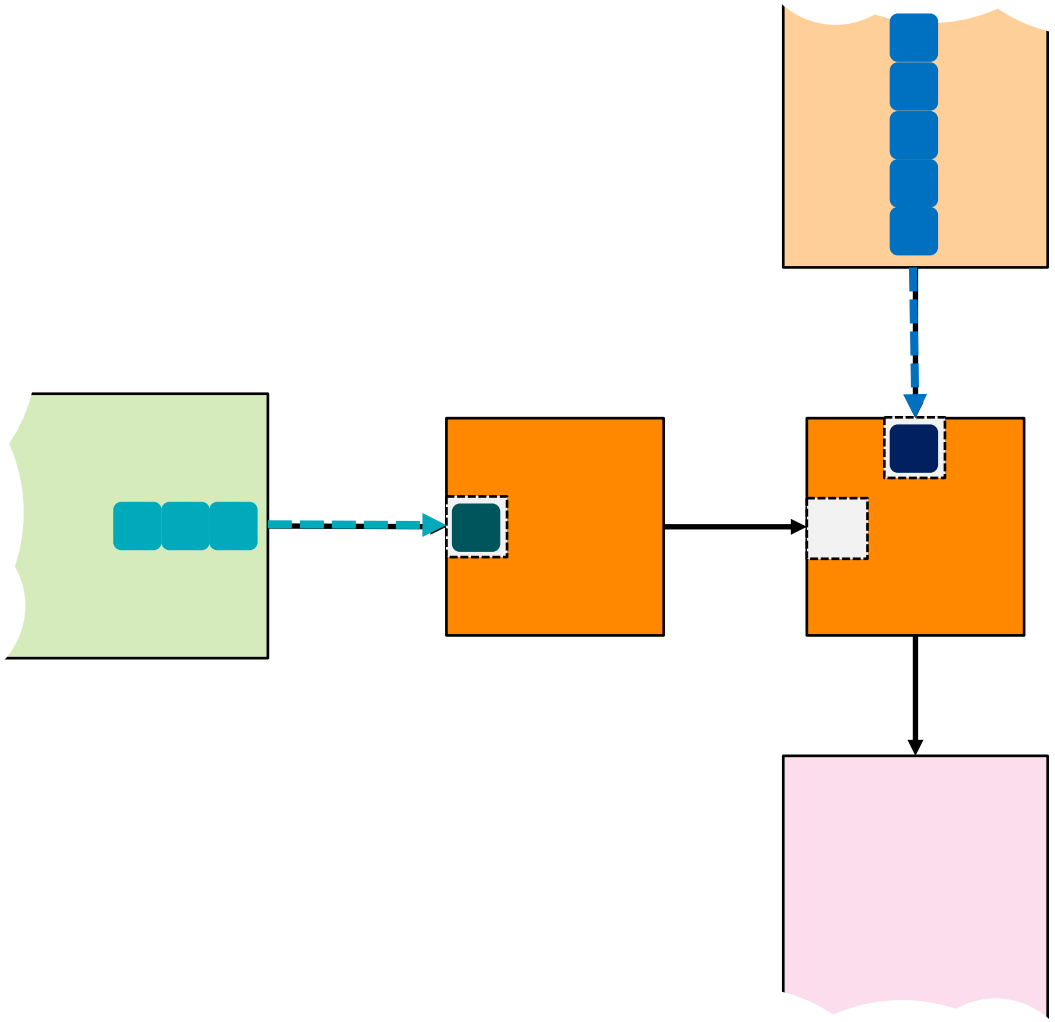
Example



Counter: 1

Wormhole Switching

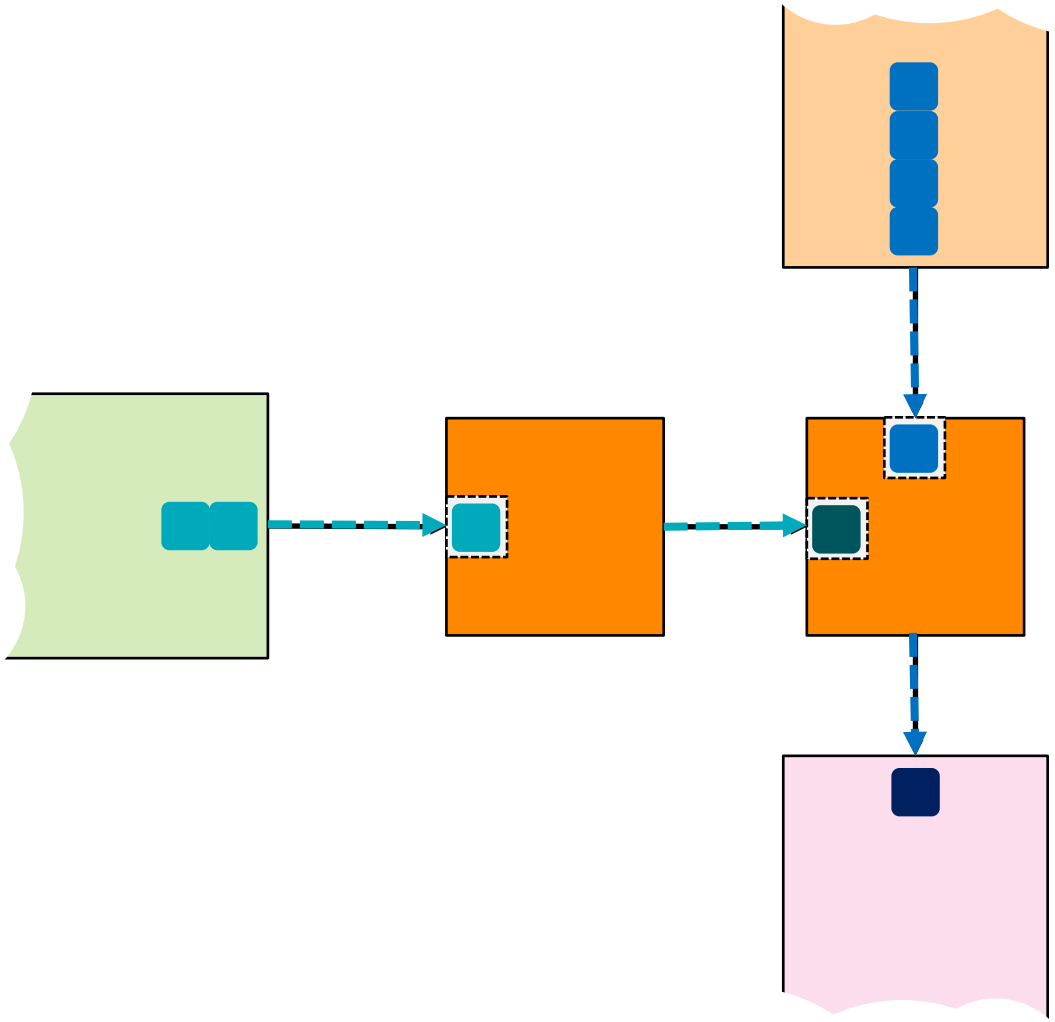
Example



Counter: 1

Wormhole Switching

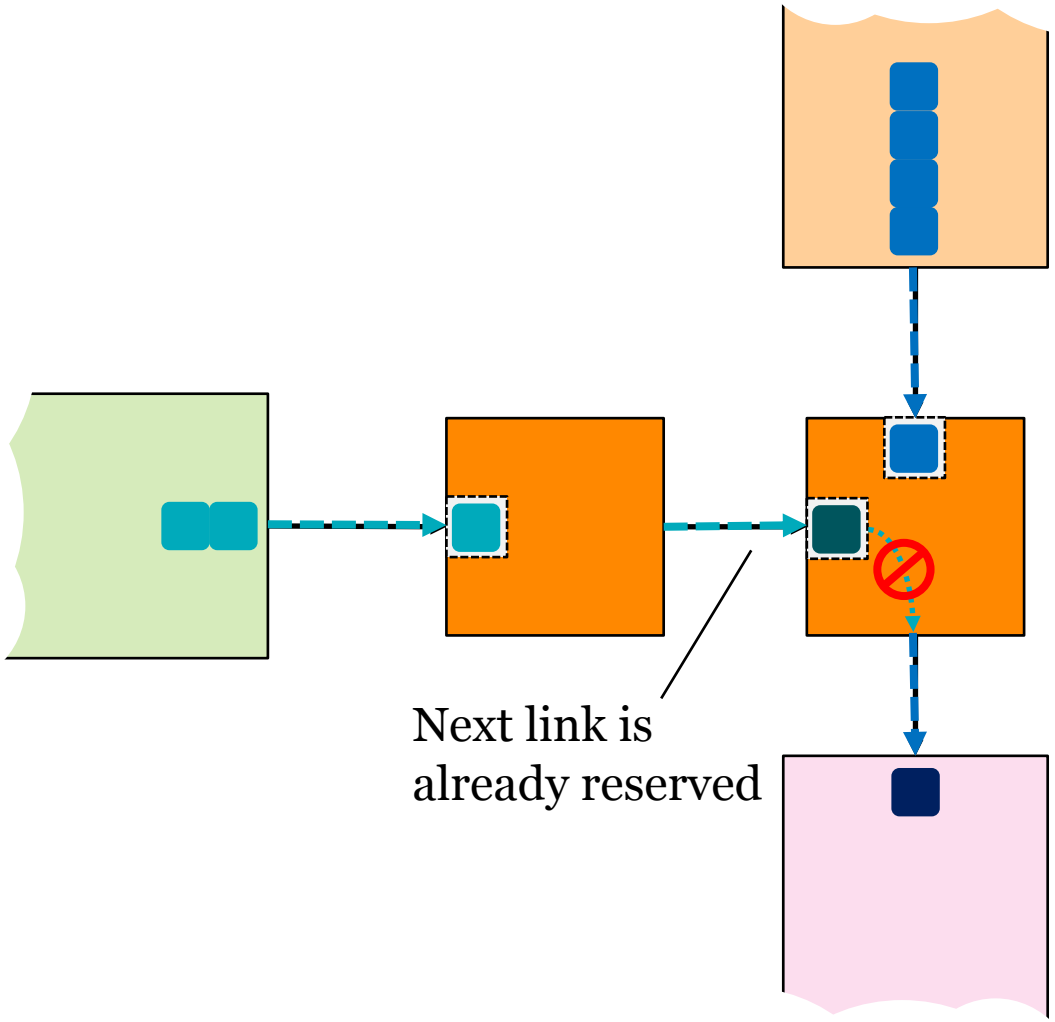
Example



Counter: 2

Wormhole Switching

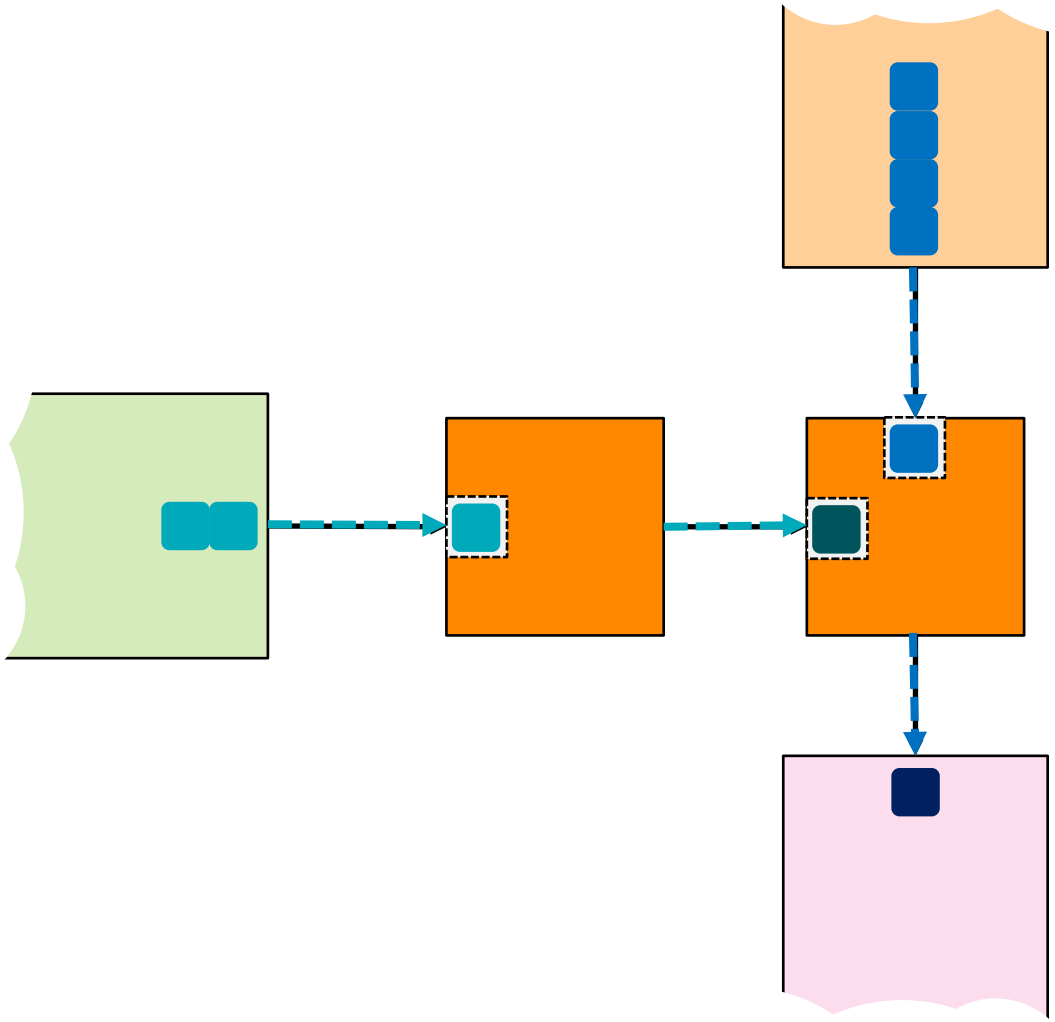
Example



Counter: 2

Wormhole Switching

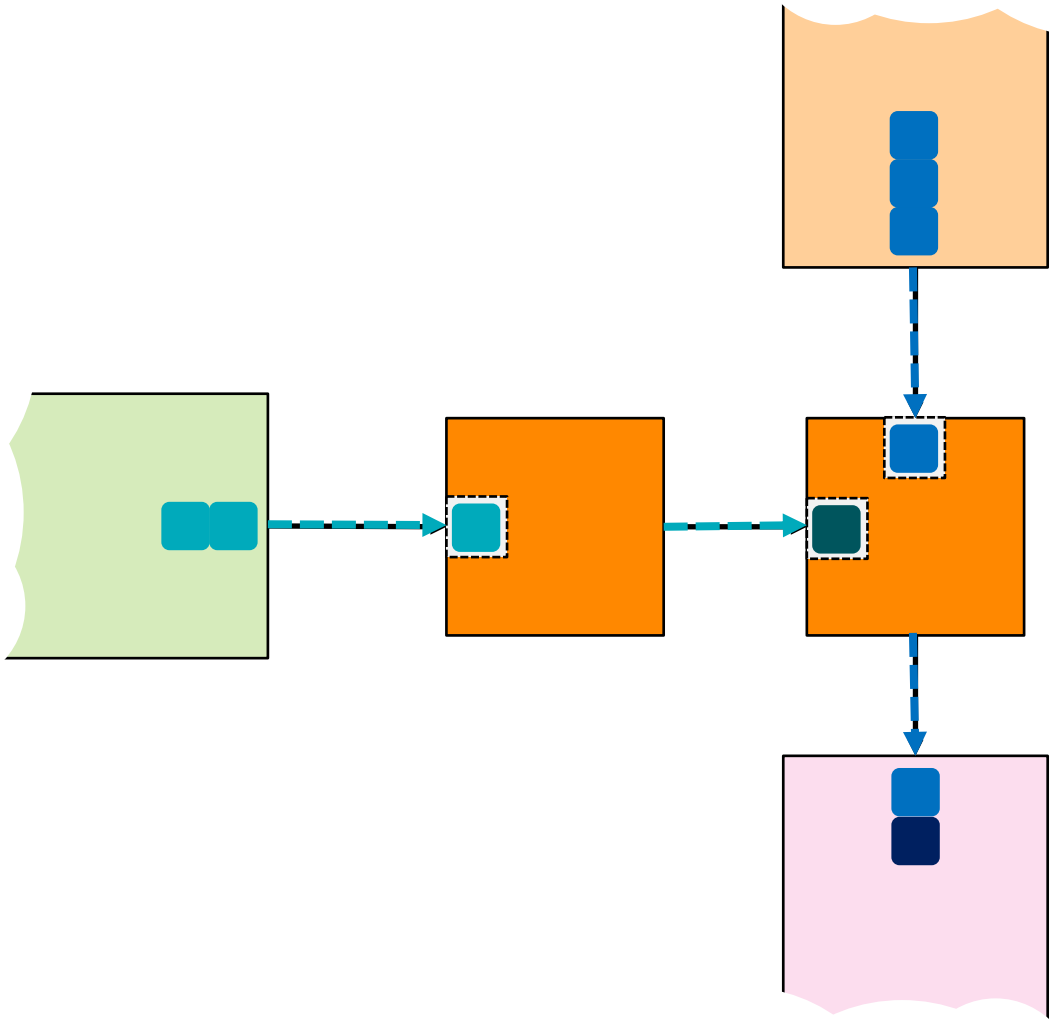
Example



Counter: 2

Wormhole Switching

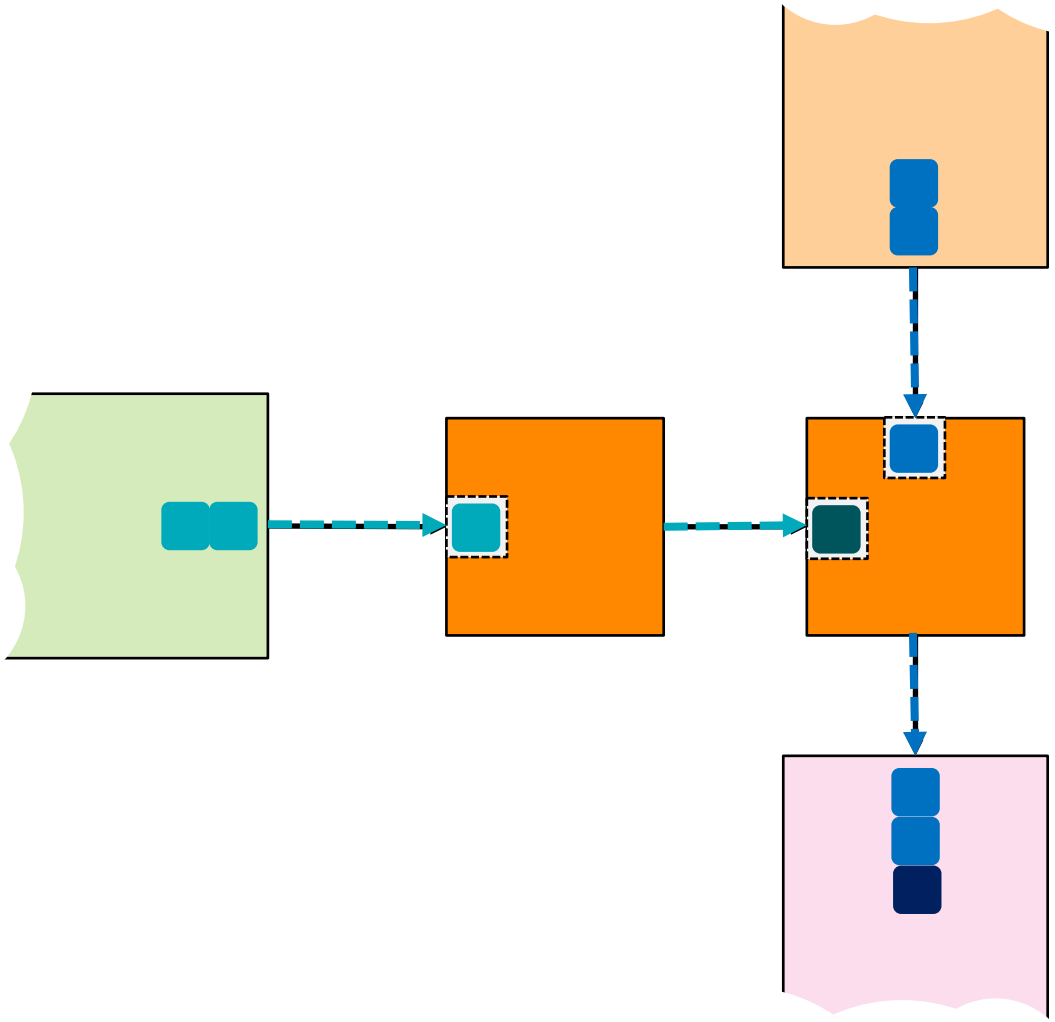
Example



Counter: 3

Wormhole Switching

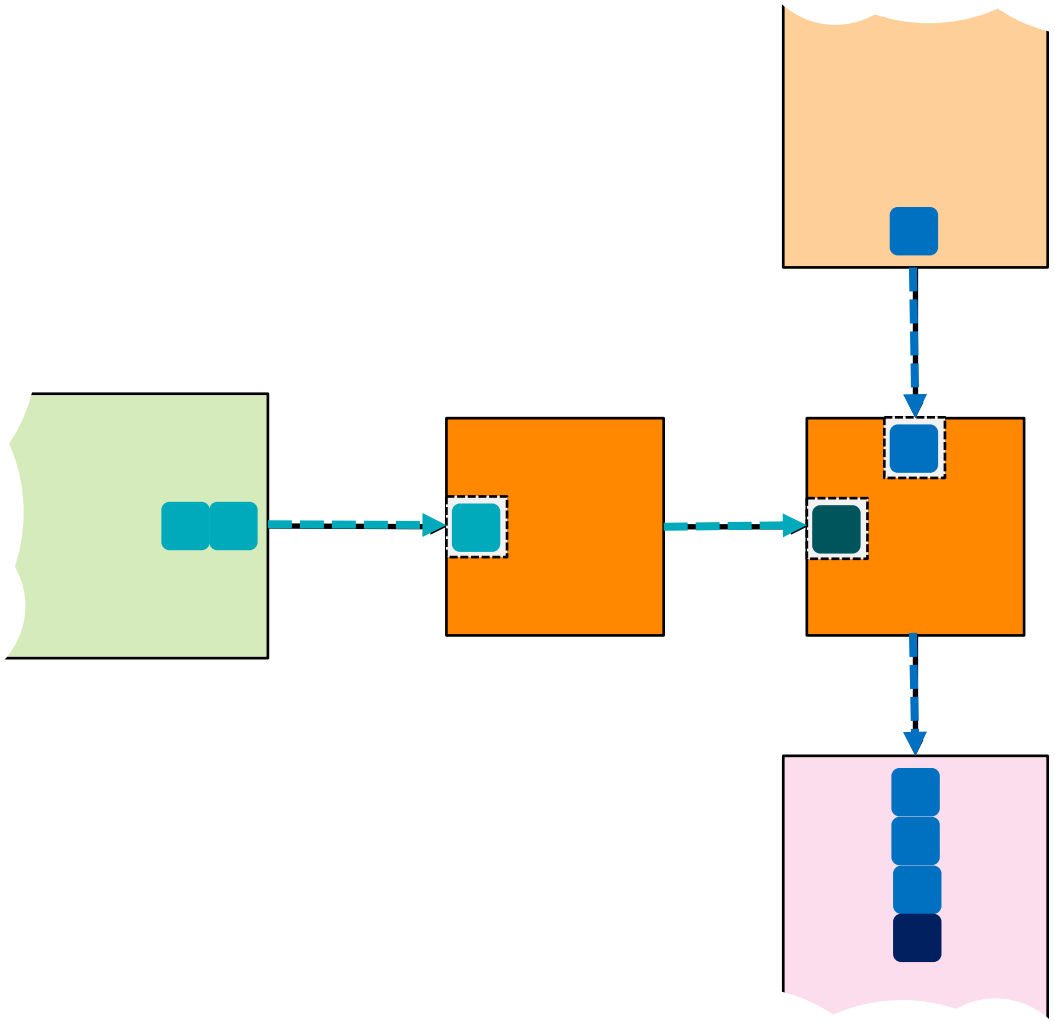
Example



Counter: 4

Wormhole Switching

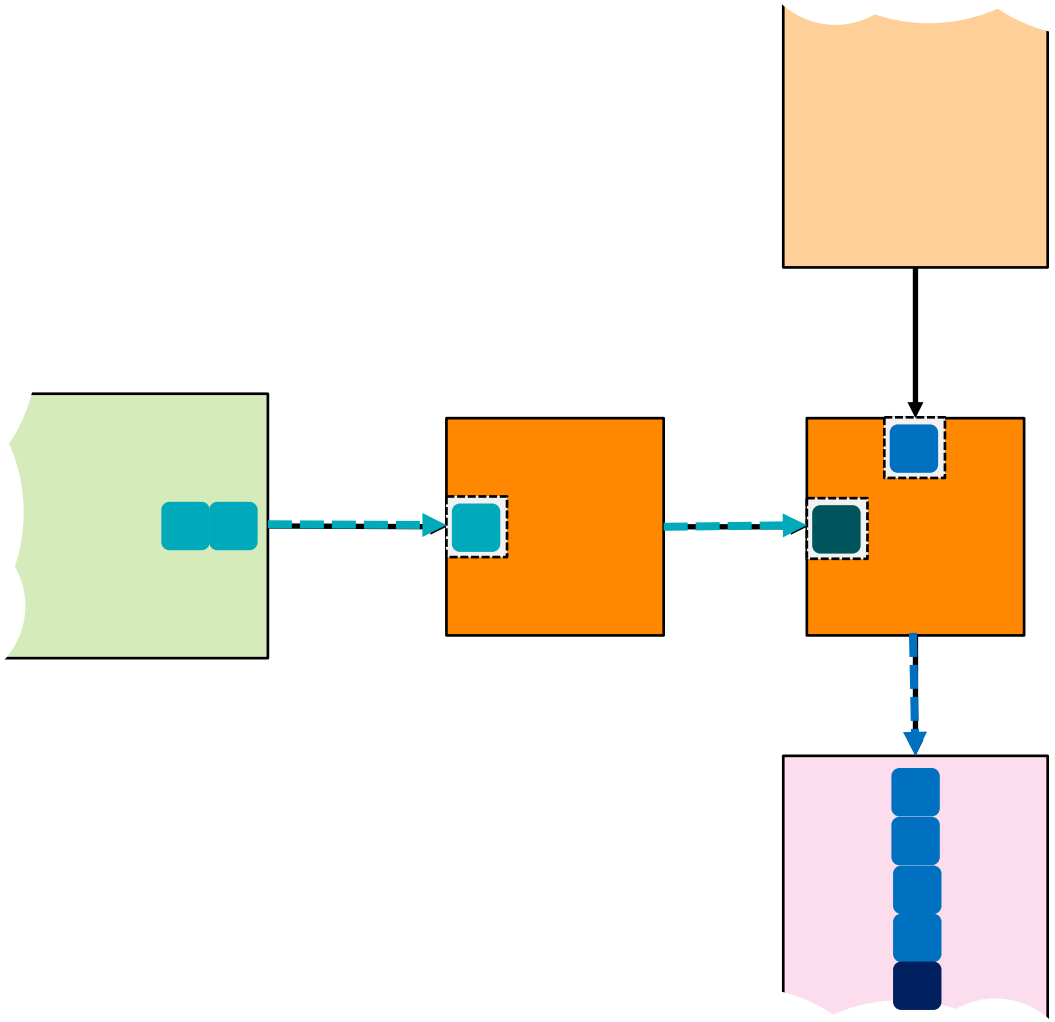
Example



Counter: 5

Wormhole Switching

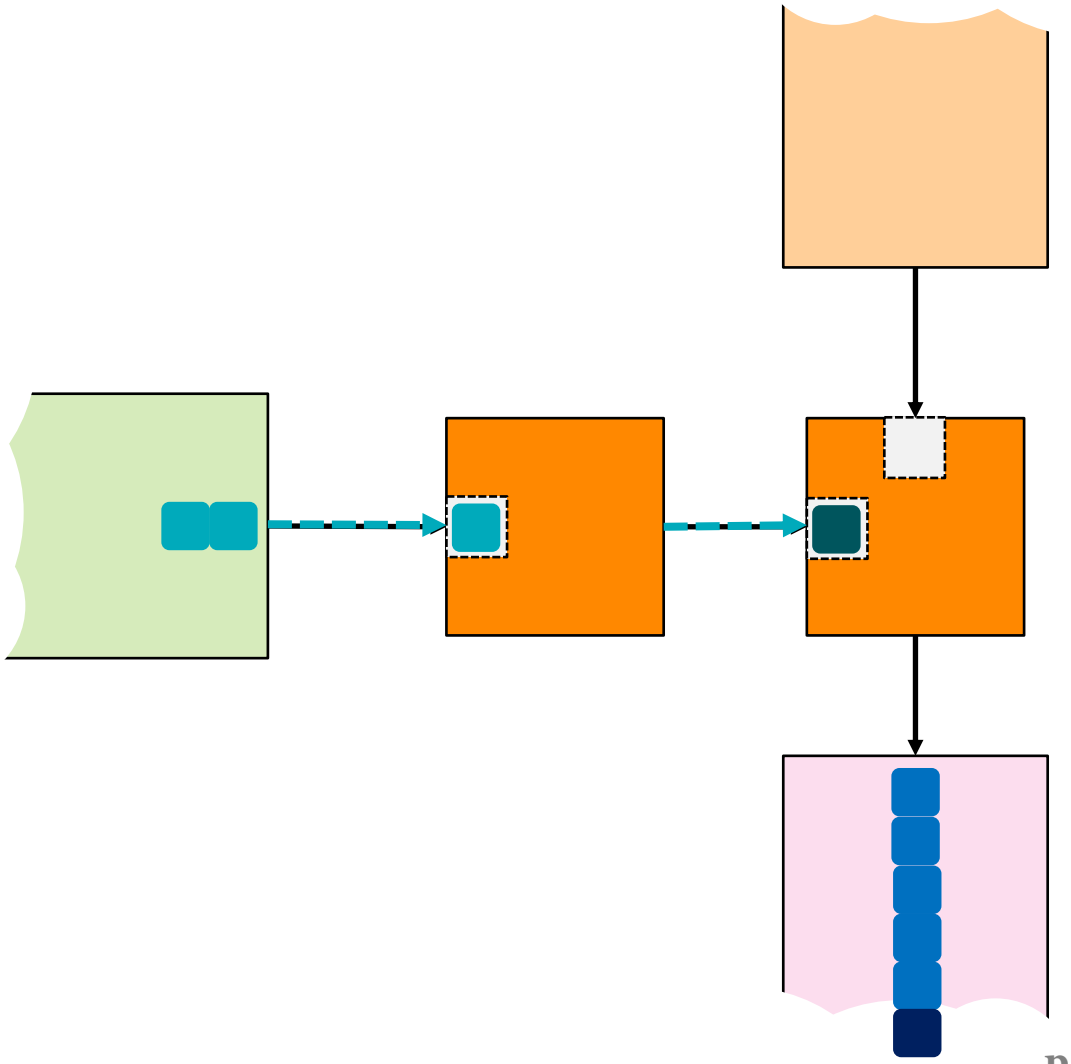
Example



Counter: 6

Wormhole Switching

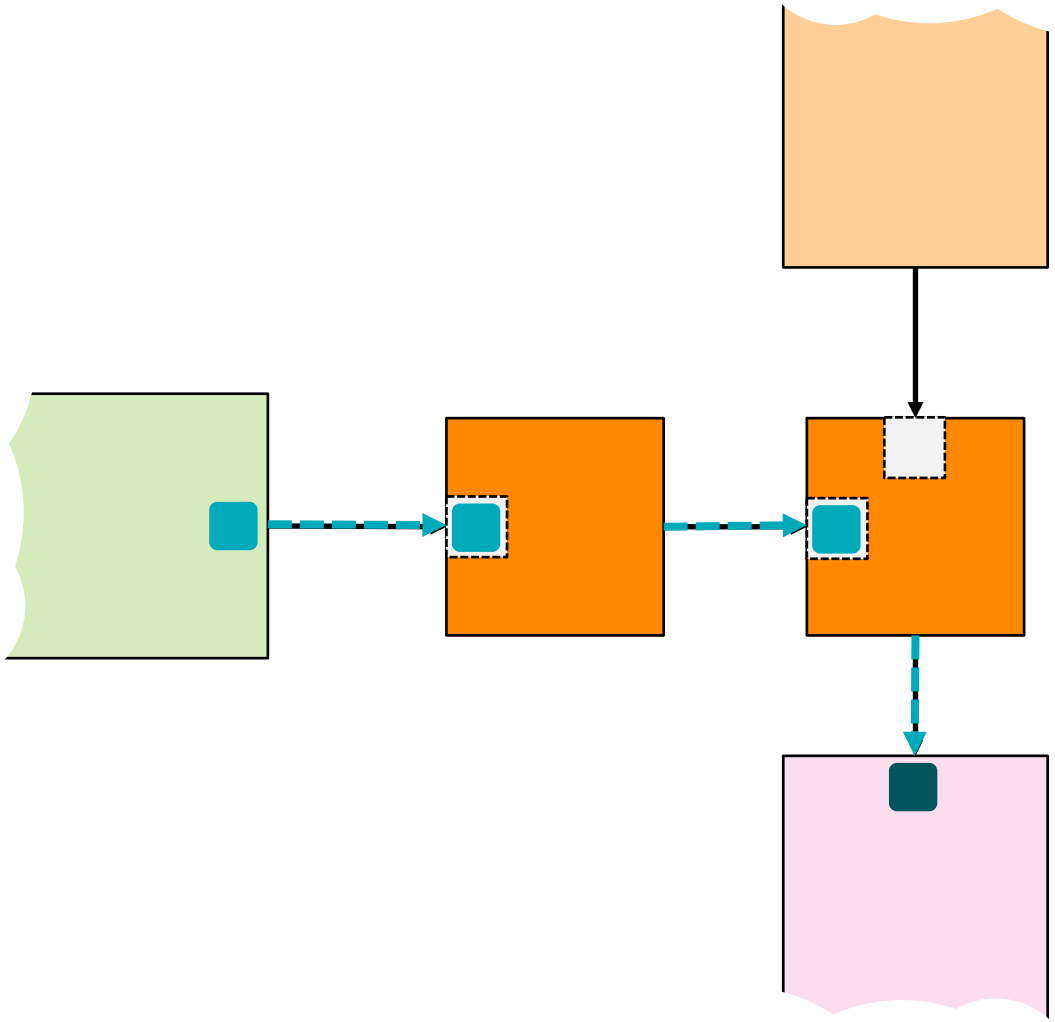
Example



Counter: 7

Wormhole Switching

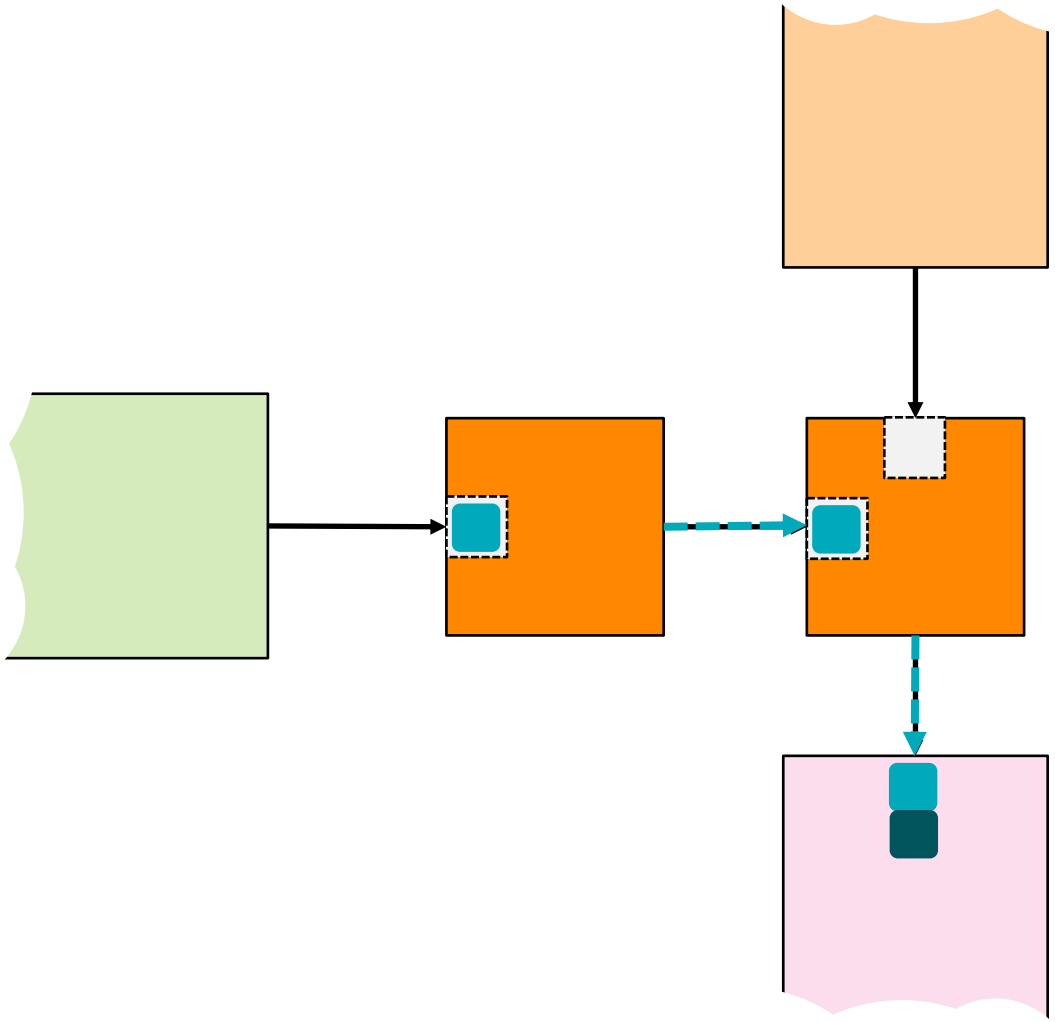
Example



Counter: 8

Wormhole Switching

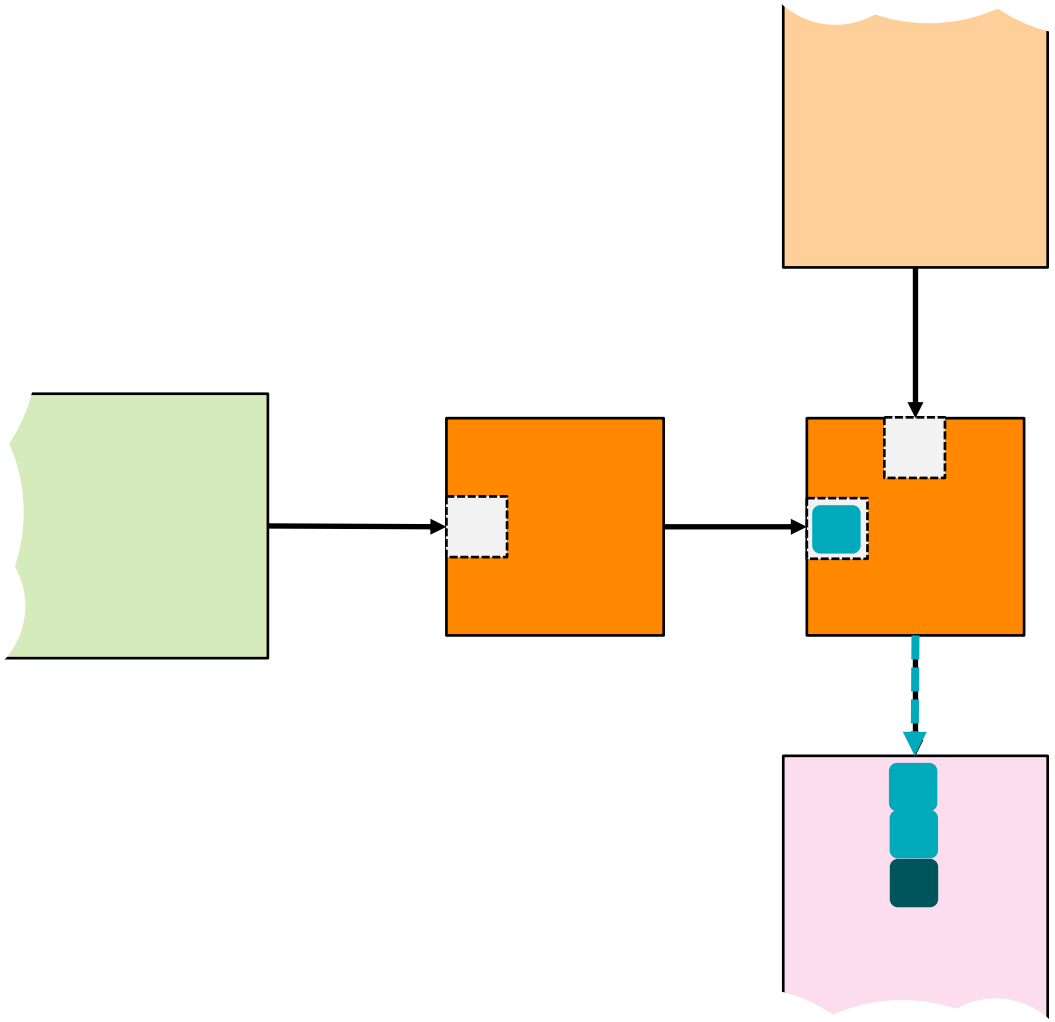
Example



Counter: 9

Wormhole Switching

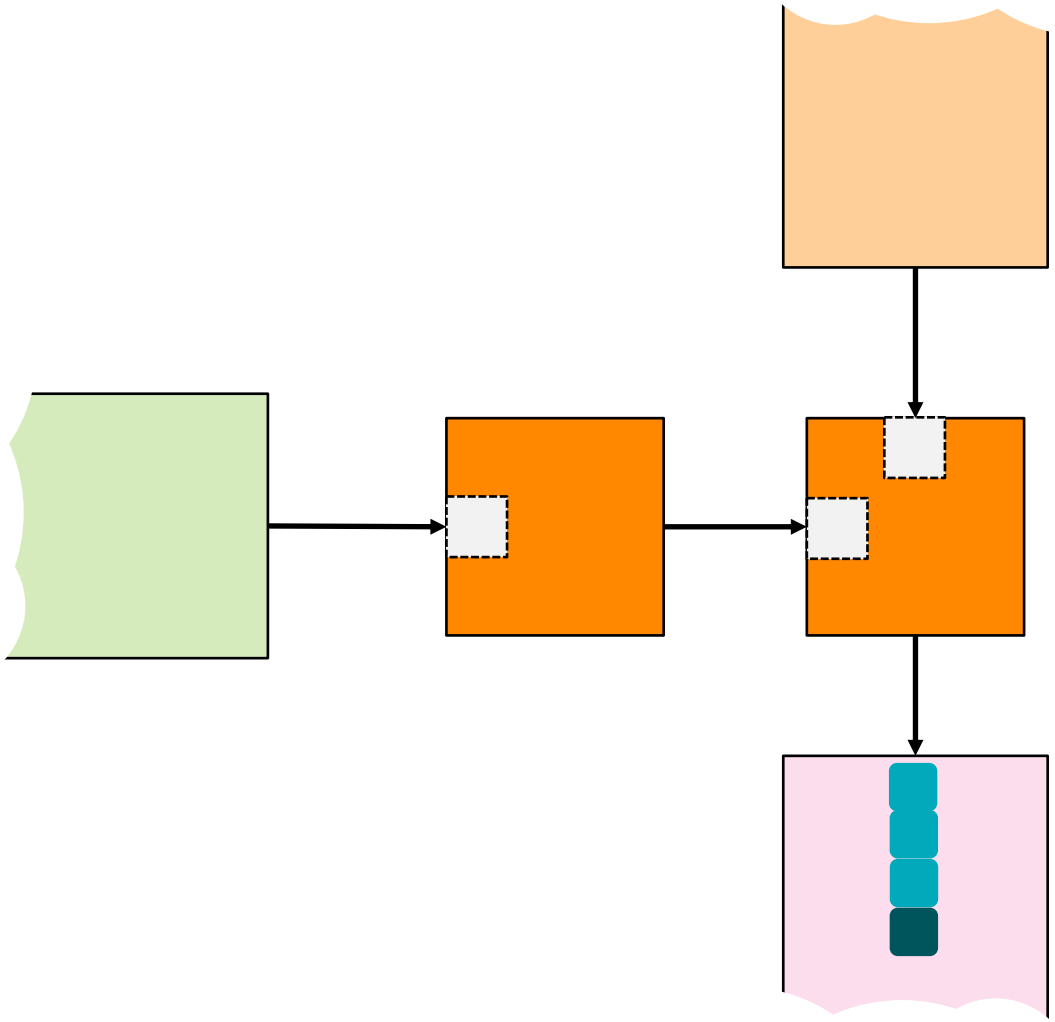
Example



Counter: 10

Wormhole Switching

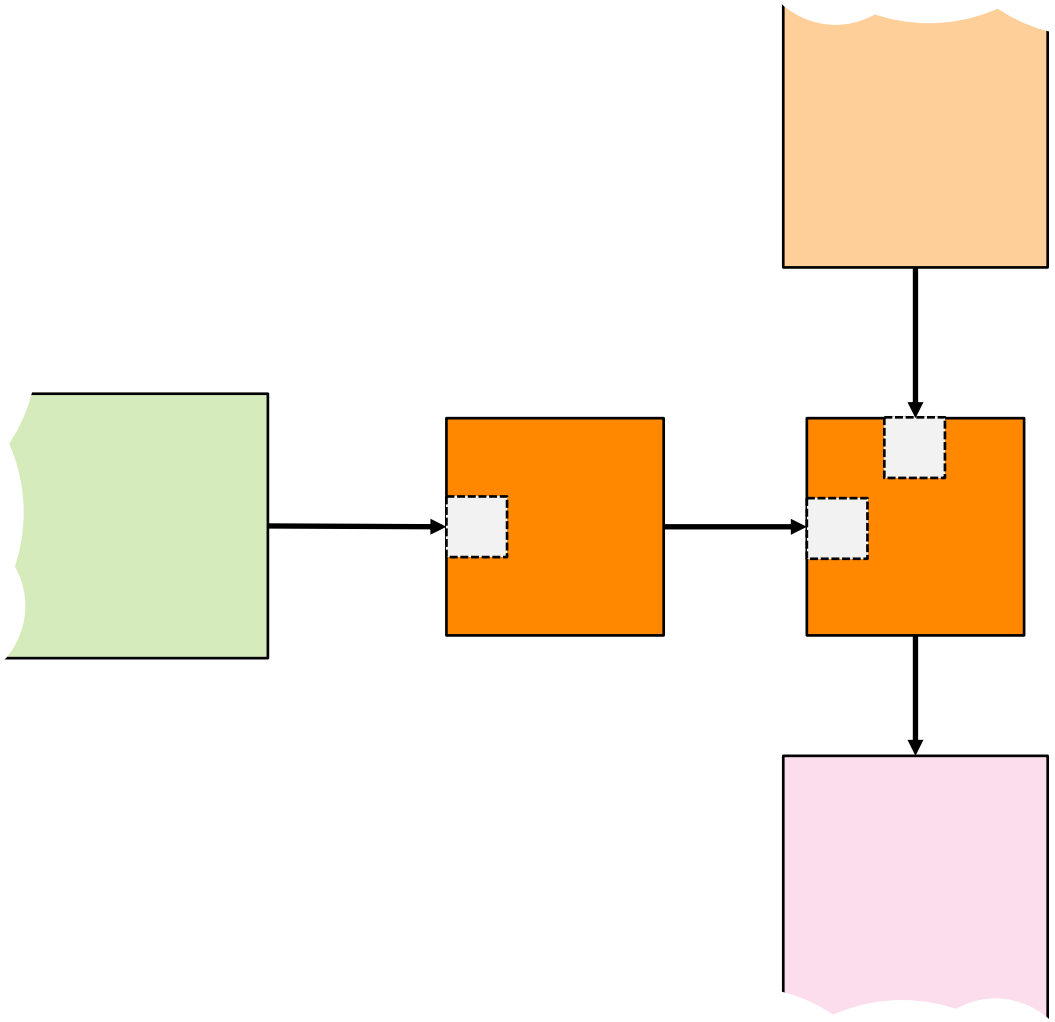
Example



Counter: 11

Wormhole Switching

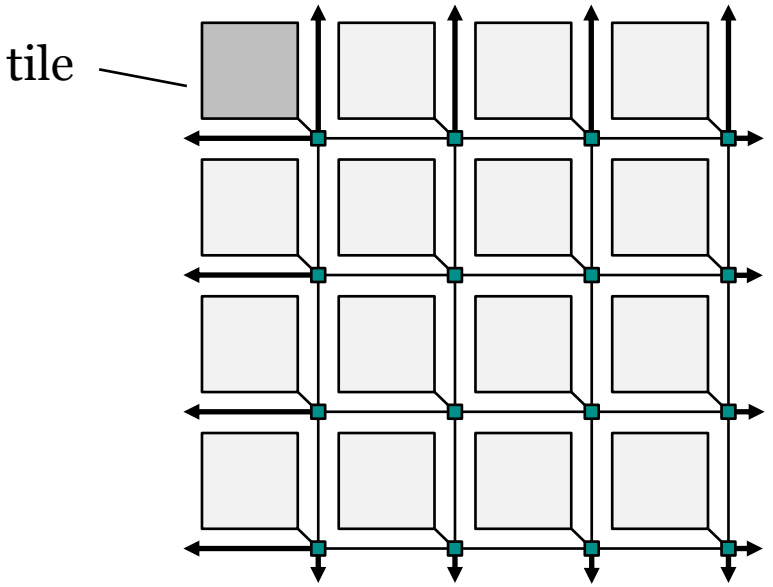
Example



Counter: 12

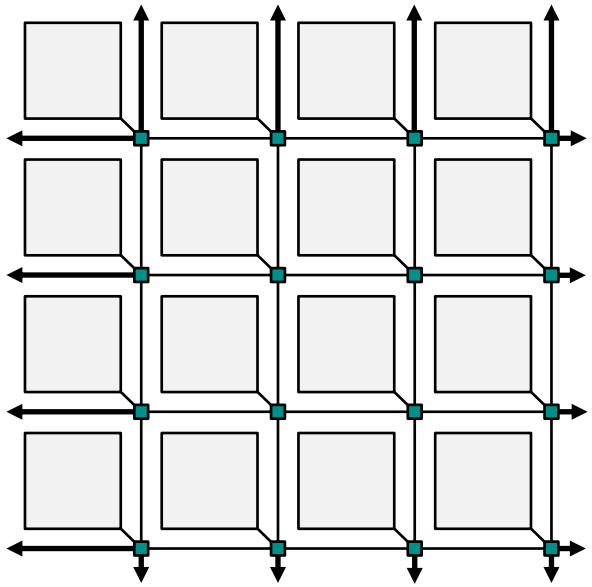
Arrangement of memory and processing units

- Elements on the many-core are arranged in tiles that are connected by the NoC



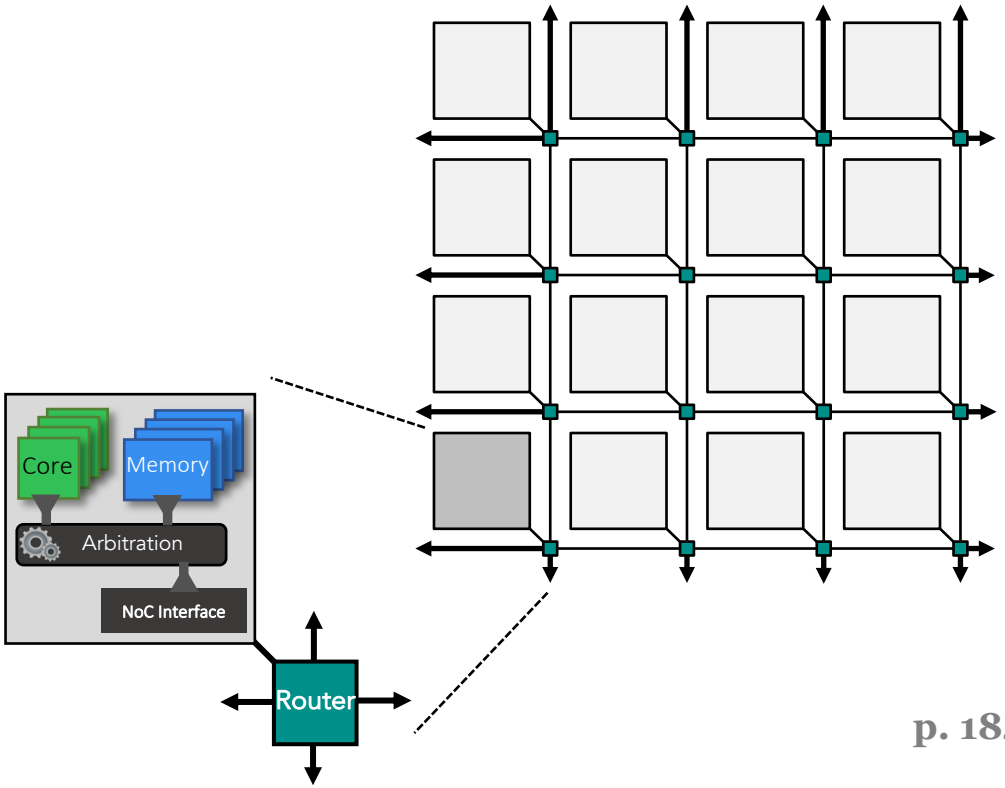
Arrangement of memory and processing units

- Elements on the many-core are arranged in tiles that are connected by the NoC
- Each tile can host:
  - Compute cores
  - Local memory
  - NoC interface



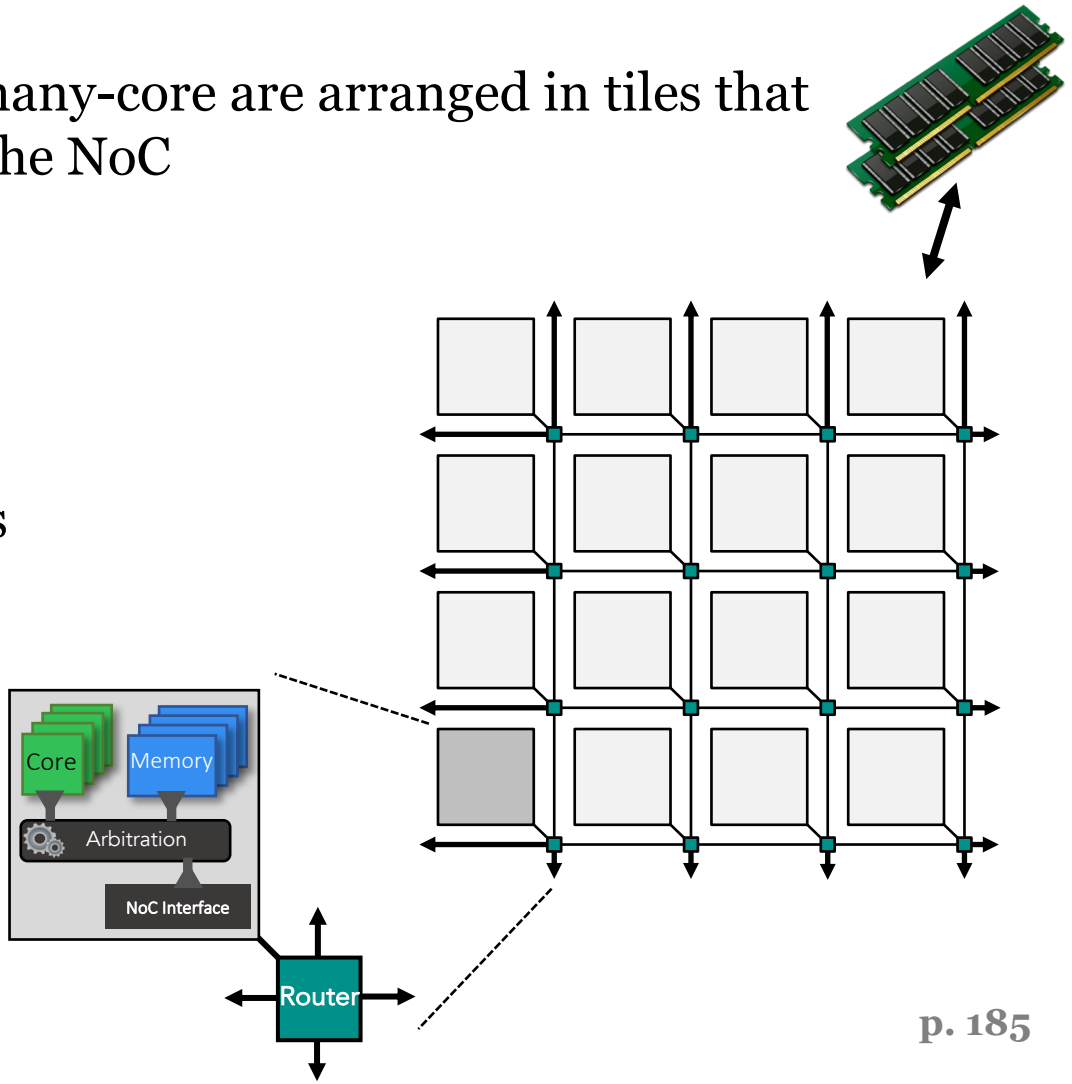
Arrangement of memory and processing units

- Elements on the many-core are arranged in tiles that are connected by the NoC
- Each tile can host:
  - Compute cores
  - Local memory
  - NoC interface



Arrangement of memory and processing units

- Elements on the many-core are arranged in tiles that are connected by the NoC
- Each tile can host:
  - Compute cores
  - Local memory
  - NoC interface
- External resources
  - Memory
  - Network
  - …



Memory Arrangement

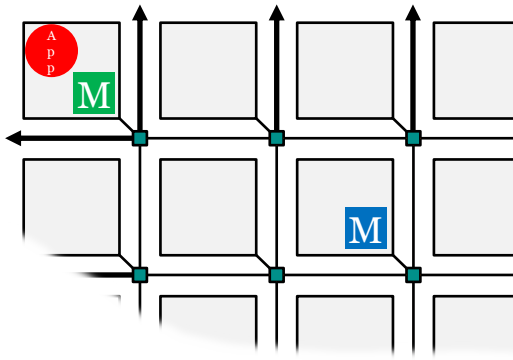
- Most many-core processors do not have global cache

Memory Arrangement

- Most many-core processors do not have global cache
- Memory is distributed on the tiles

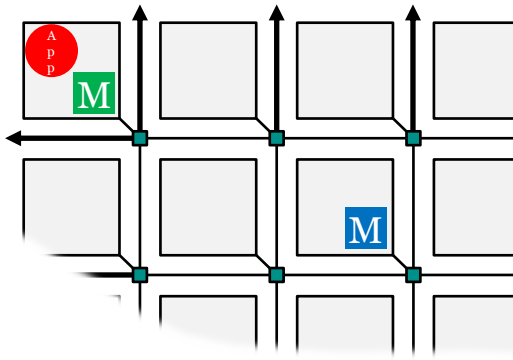
Memory Arrangement

- Most many-core processors do not have global cache
- Memory is distributed on the tiles
- Locality becomes important
  - Access to local memory is cheap
  - Access to memory on an other tile is expensive



Memory Arrangement

- Most many-core processors do not have global cache
- Memory is distributed on the tiles
- Locality becomes important
  - Access to local memory is cheap
  - Access to memory on an other tile is expensive
- Access to memory on an other tile or external to the chip is done over the NoC
  - Often multiple instances of the NoC for request and response messages

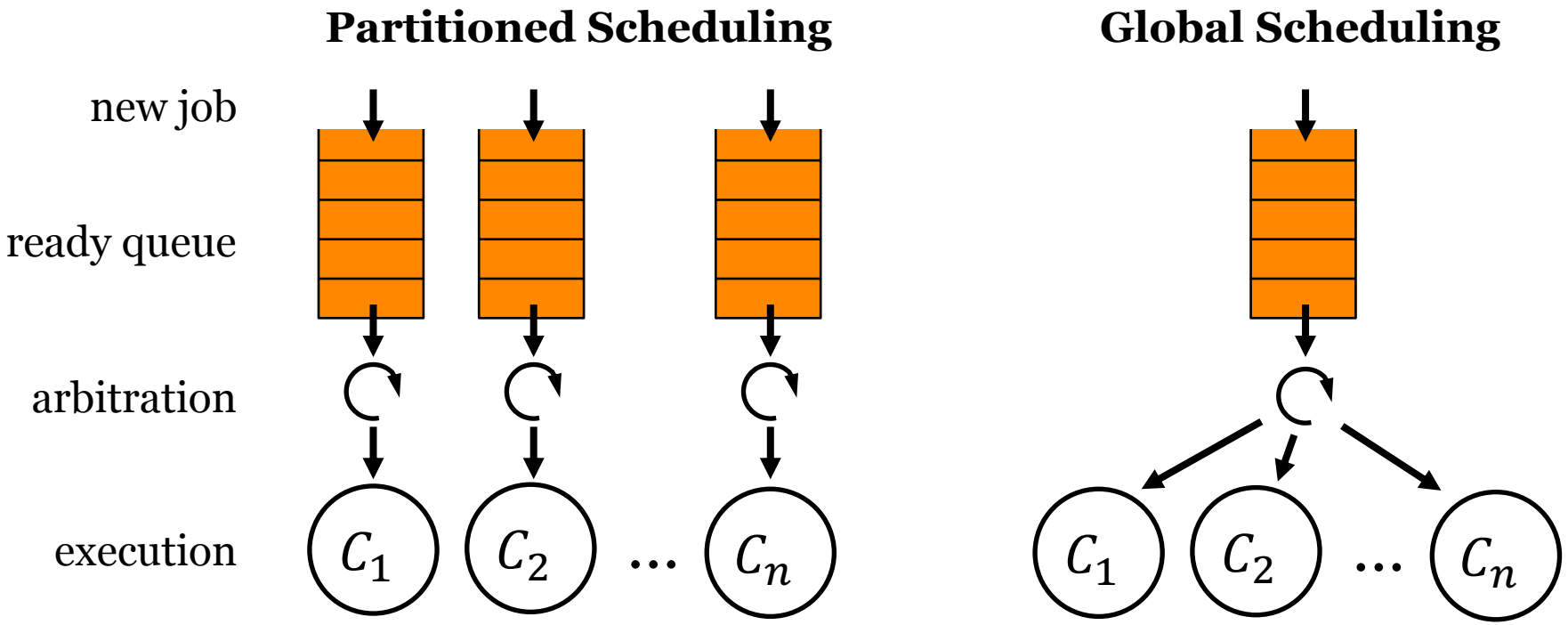


Execution Models

- Which scheduling models do you know for multiprocessors?

Execution Models

- Which scheduling models do you know for multiprocessors?



Execution Models

- Why is partitioned scheduling better suited for many-core processors than global scheduling?

Execution Models

- Why is partitioned scheduling better suited for many-core processors than global scheduling?
- Resources that are shared amongst all cores lead to large blocking times in global scheduling
- Analysing a complete system becomes hard due to the system complexity

Execution Models

- Why is partitioned scheduling better suited for many-core processors than global scheduling?
- Resources that are shared amongst all cores lead to large blocking times in global scheduling
- Analysing a complete system becomes hard due to the system complexity
- If each core or tile is independent the blocking due to shared resources is reduced
- Each tile can be analysed independently
  - Many techniques developed for distributed systems can be used

Execution Models

- Why is partitioned scheduling better suited for many-core processors than global scheduling?
- Resources that are shared amongst all cores lead to large blocking times in global scheduling
- Analysing a complete system becomes hard due to the system complexity
- If each core or tile is independent the blocking due to shred resources is reduced
- Each tile can be analysed independently
  - Many techniques developed for distributed systems can be used

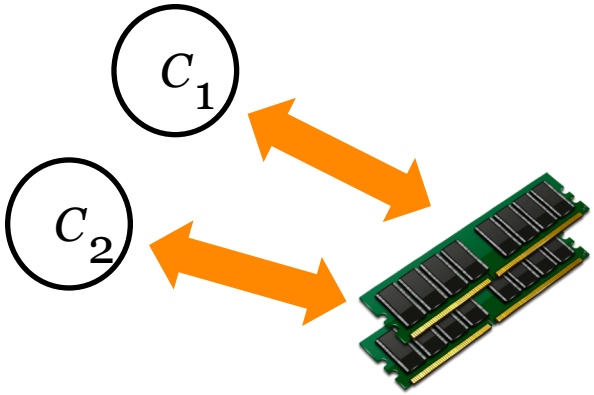
→ Divide and conquer approach

Challenges

- Increasing number of clients → Increasing potential for conflicts

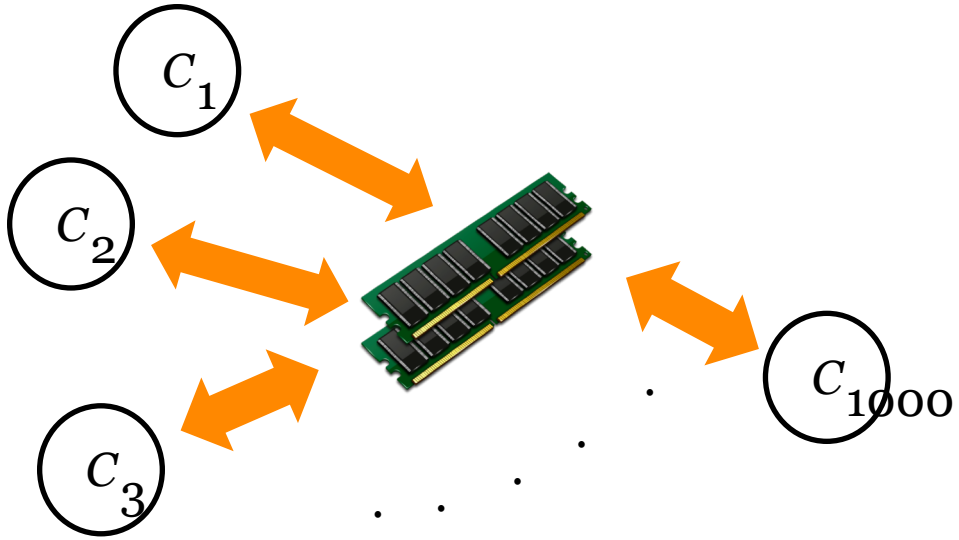
Challenges

- Increasing number of clients → Increasing potential for conflicts



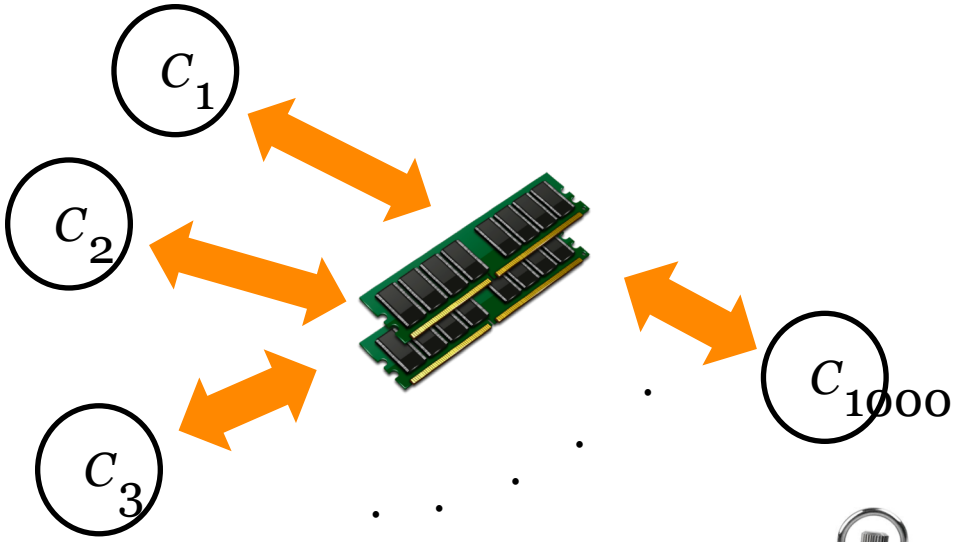
Challenges

- Increasing number of clients → Increasing potential for conflicts

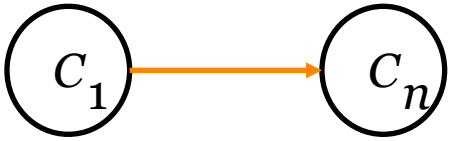


Challenges

- Increasing number of clients → Increasing potential for conflicts

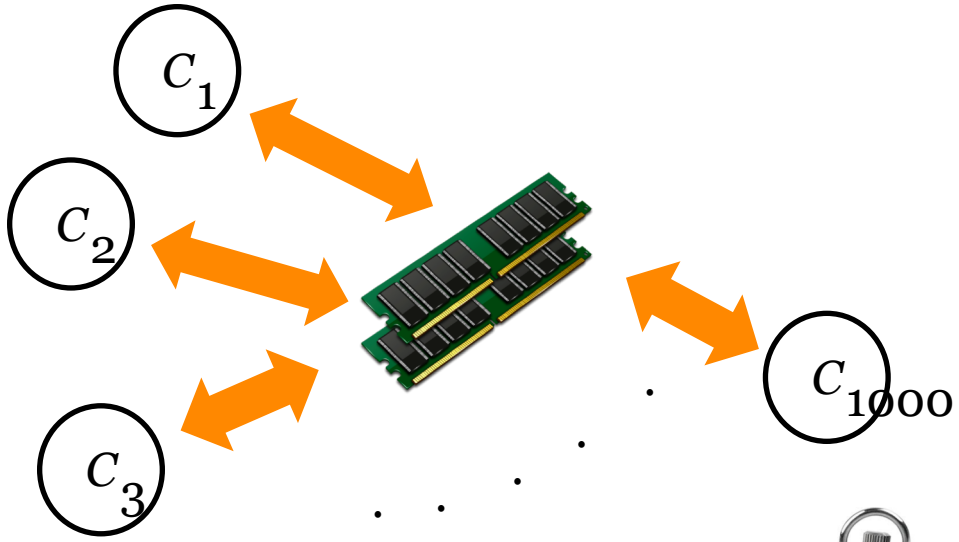


- Locality is important → Influences the message delays



Challenges

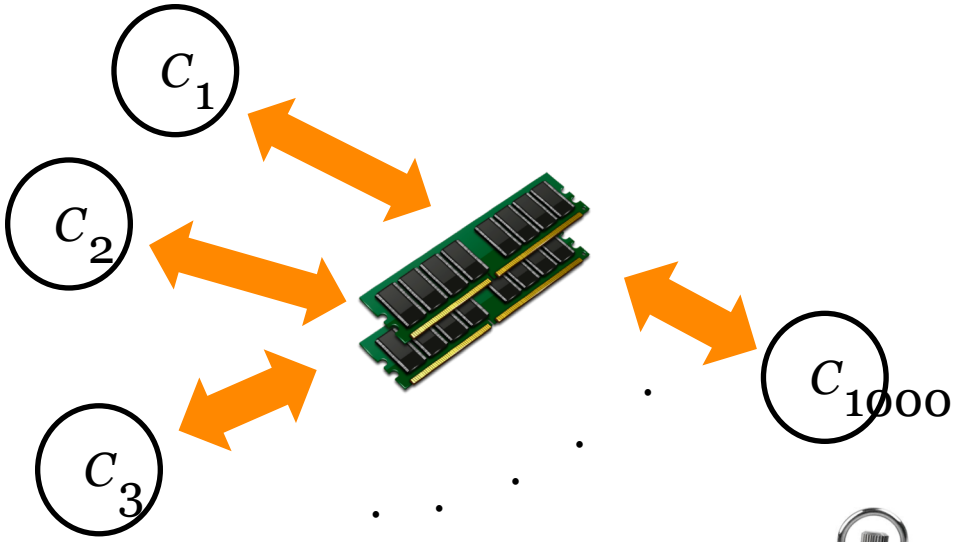
- Increasing number of clients → Increasing potential for conflicts



- Locality is important → Influences the message delays

Challenges

- Increasing number of clients → Increasing potential for conflicts

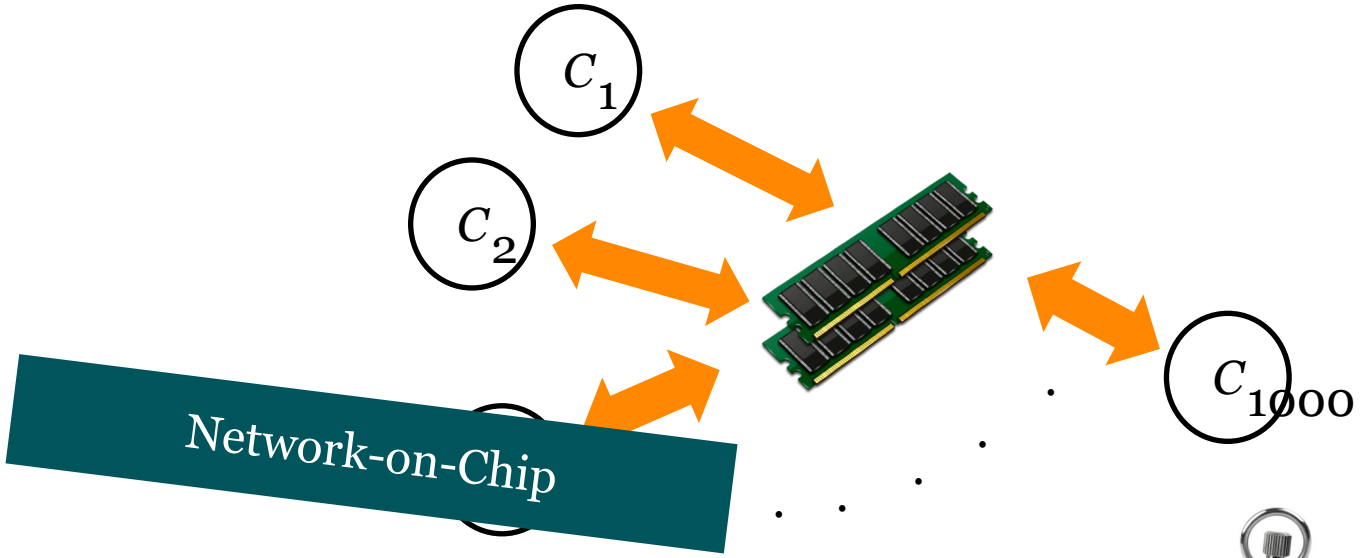


- Locality is important → Influences the message delays

Mapping problem → binpacking → Complicated!

Challenges

- Increasing number of clients → Increasing potential for conflicts

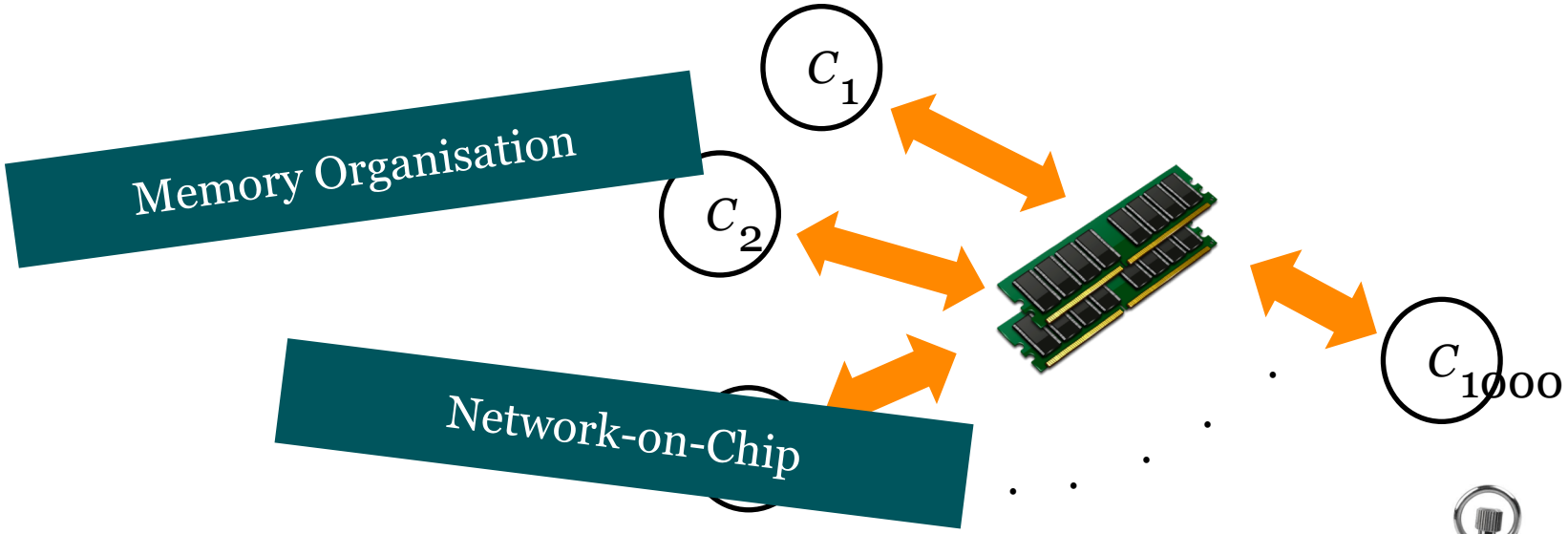


- Locality is important → Influences the message delays

Mapping problem → binpacking → Complicated!

Challenges

- Increasing number of clients → Increasing potential for conflicts

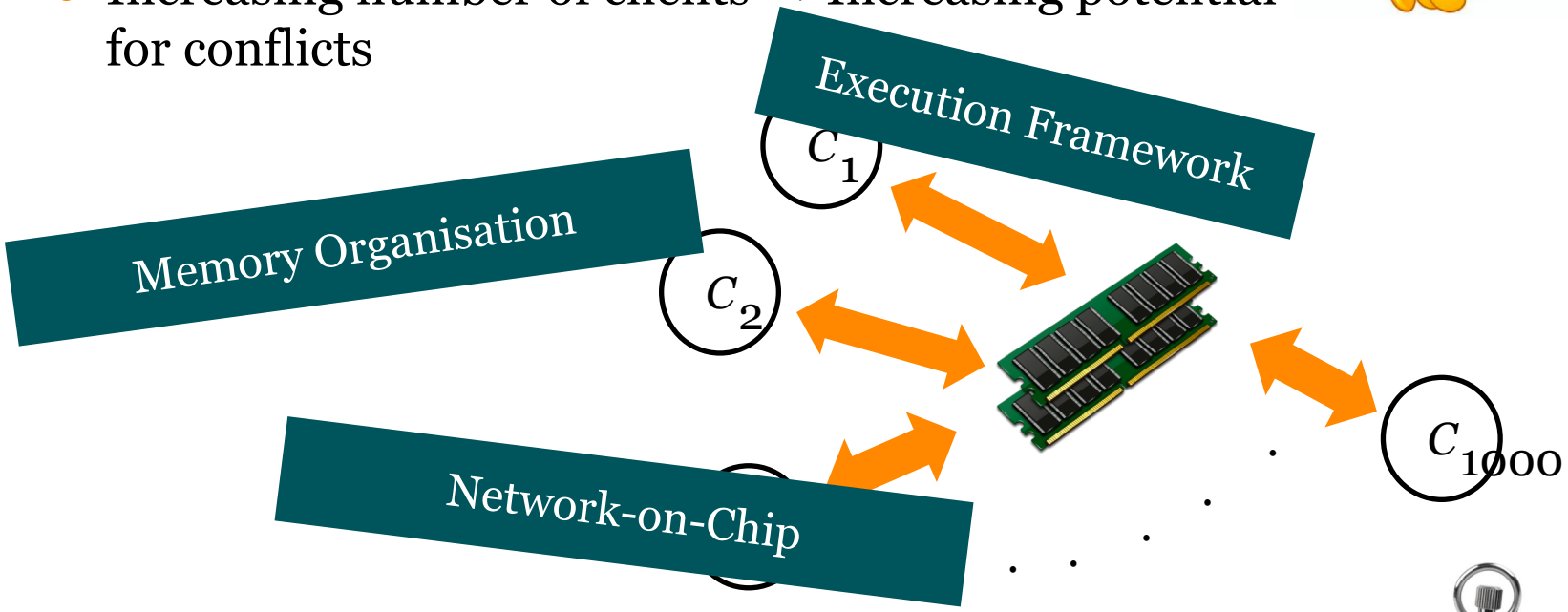


- Locality is important → Influences the message delays

Mapping problem → binpacking → Complicated!

Challenges

- Increasing number of clients → Increasing potential for conflicts

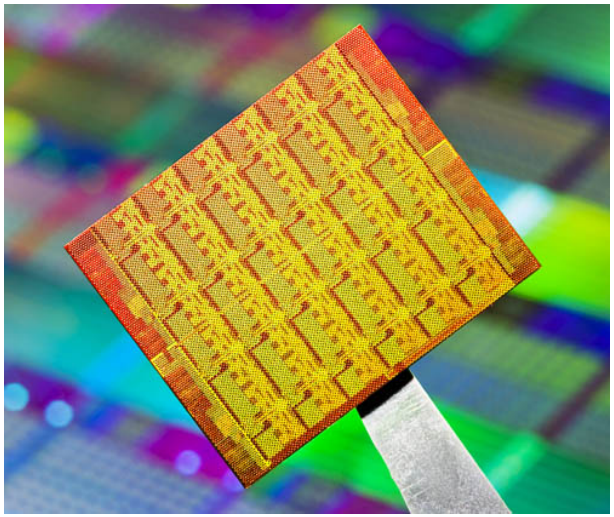


- Locality is important → Influences the message delays

Mapping problem → binpacking → Complicated!

”COTS” Many-Core Processors

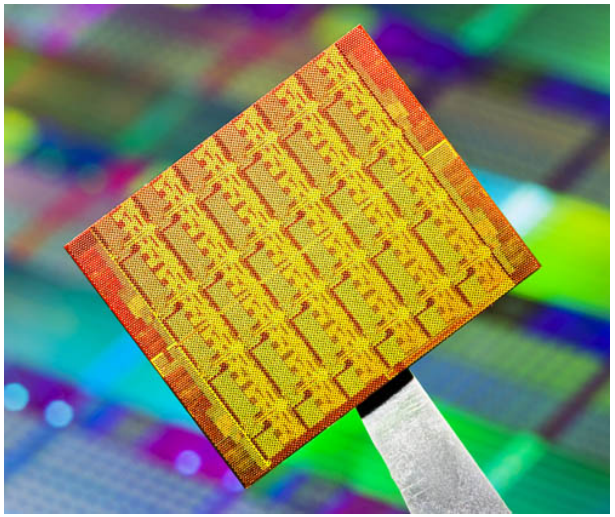
- Several companies with many-core processors on the market (Tilera, Kalray, Adapteva, Intel, …)



COTS = Commercial Of The Shelf

”COTS” Many-Core Processors

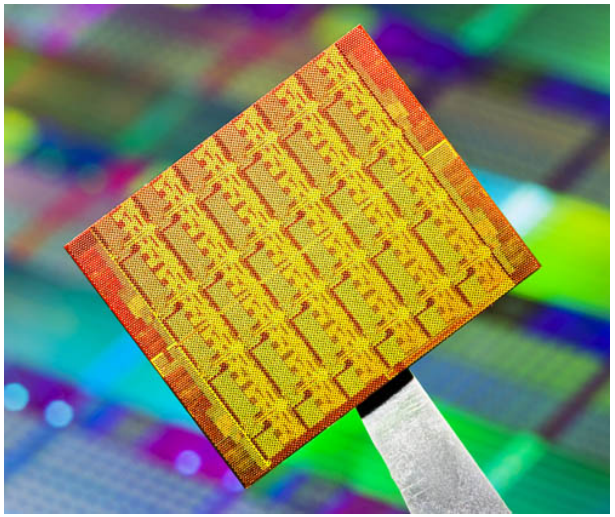
- Several companies with many-core processors on the market (Tilera, Kalray, Adapteva, Intel, …)
  - What is on a tile?
    - Core + local memory
    - 2 cores + cache
    - 16 cores + cache + local memory
    - …



COTS = Commercial Of The Shelf

”COTS” Many-Core Processors

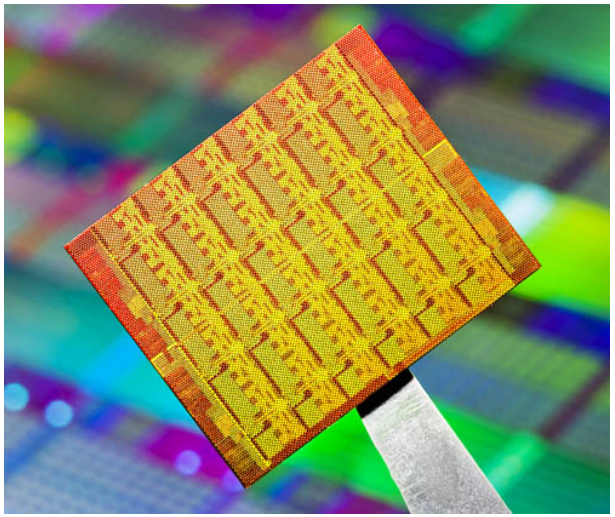
- Several companies with many-core processors on the market (Tilera, Kalray, Adapteva, Intel, …)
  - What is on a tile?
    - Core + local memory
    - 2 cores + cache
    - 16 cores + cache + local memory
    - …
  - How are the tiles connected?
    - 2D-mesh based NoC
    - 2D-wrapped-around torus NoC
    - With / without flow control
    - 1 / 2 / 4 / … parallel networks
    - Round Robin arbitrations / Virtual Channels
    - …



COTS = Commercial Of The Shelf

”COTS” Many-Core Processors

- Several companies with many-core processors on the market (Tilera, Kalray, Adapteva, Intel, …)
  - What is on a tile?
    - Core + local memory
    - 2 cores + cache
    - 16 cores + cache + local memory
    - …
  - How are the tiles connected?
    - 2D-mesh based NoC
    - 2D-wrapped-around torus NoC
    - With / without flow control
    - 1 / 2 / 4 / … parallel networks
    - Round Robin arbitrations / Virtual Channels
    - …
  - Stand alone usage?
    - Yes / no



COTS = Commercial Of The Shelf

”COTS” Many-Core Processors

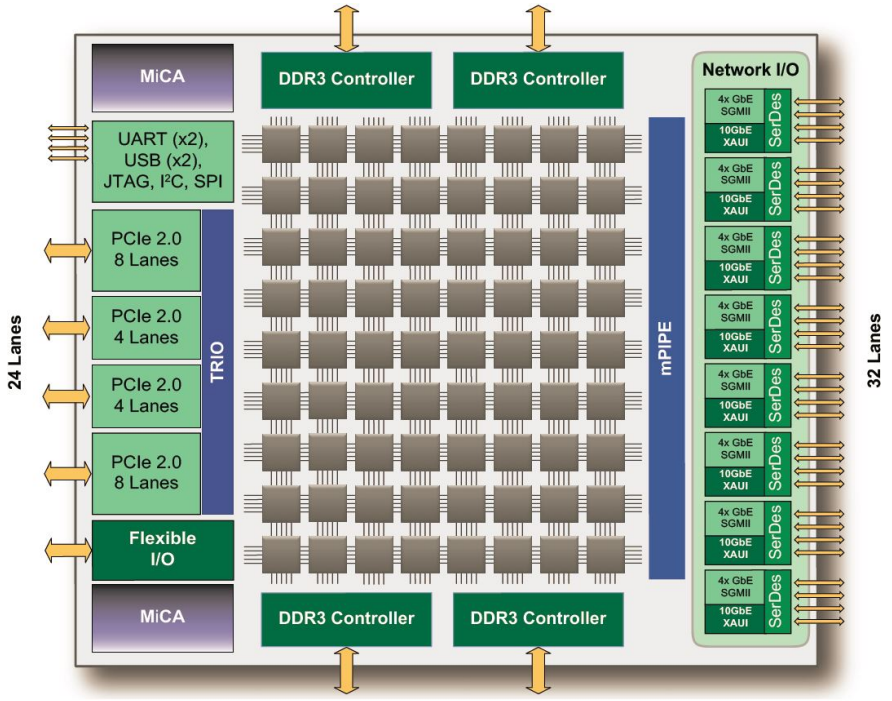
- Several companies with many-core processors on the market (Tilera, Kalray, Adapteva, Intel, …)
  - What is on a tile?
    - Core + local memory
    - 2 cores + cache
    - 16 cores + [illegible]
    - [illegible]-around torus NoC
    - With / without flow control
    - 1 / 2 / 4 / … parallel networks
    - Round Robin arbitrations / Virtual Channels
    - …
  - Stand alone usage?
    - Yes / no

Large variety of available hardware architectures!
With their own advantages / disadvantages!

COTS = Commercial Of The Shelf

Tilera Tile GX

- 64-bit RISC cores
- 3-level cache coherent shared memory
- Several NoC implemented
  - RR arbitration
- Hardwall technology
- Non uniform memory access
- Available in several configurations
  - 9 / 16 / 36 / 72 cores
- Can be used as standalone processor



TILE-Gx8072 Processor Block Diagram

Kalray MPPA 256

- Several Cores
  - 256 cores
- Clustered architecture (each 16 compute cores)
- 2D-wrapped-around torus NoC
- Two NoC implemented
  - Data NoC
  - Control NoC
  - Flow control on the source nodes
- Each I/O subsystem hosts a quadcore CPU which can run Linux in SMP mode

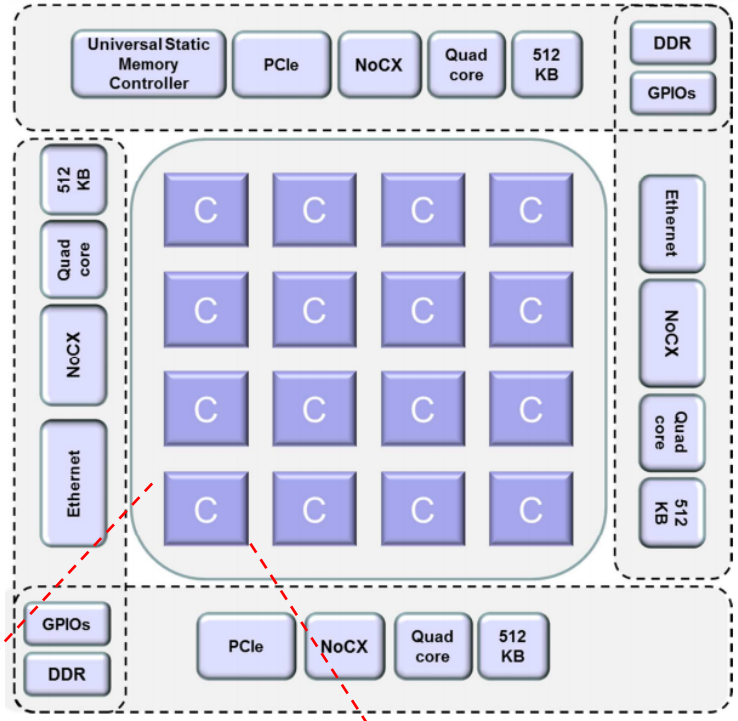
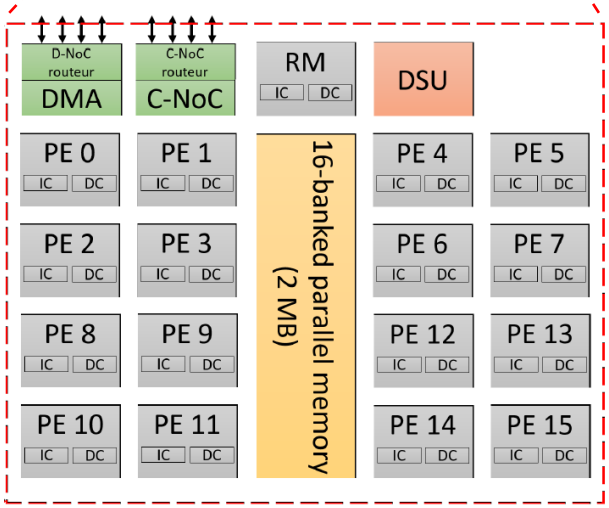
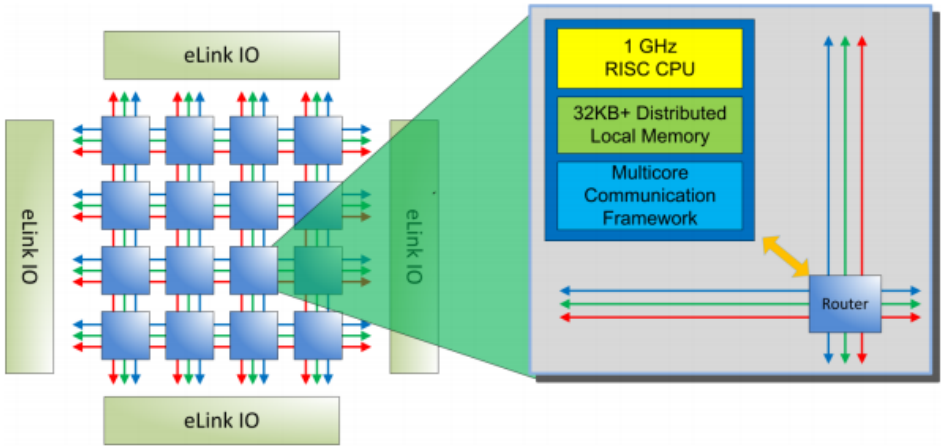


Figure 1 – MPPA®-256 block diagram

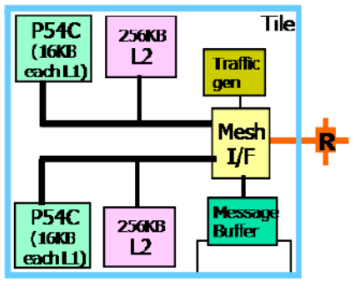


Adapteva Epiphany

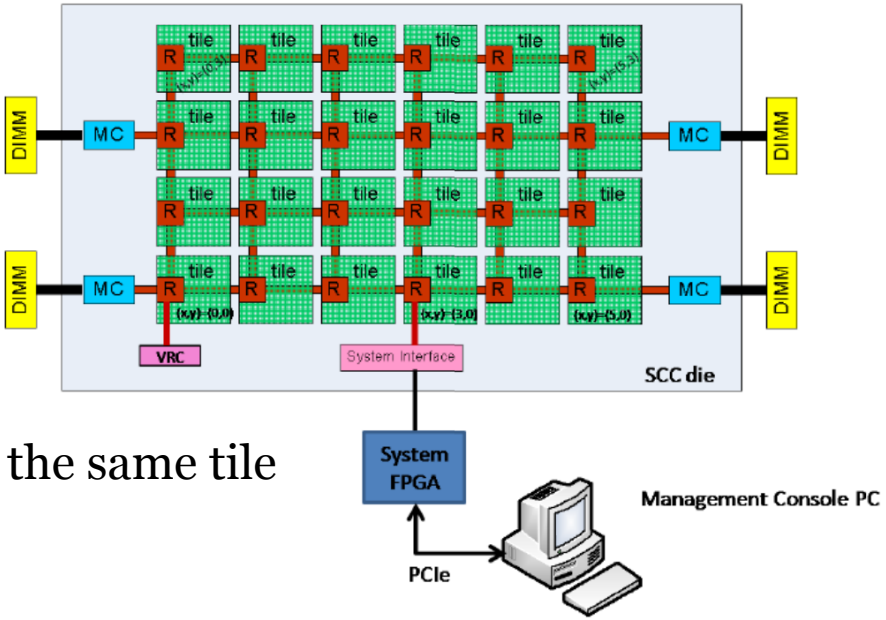
- 32-bit RISC CPU cores
- Distributed memory model
  - 32KB local memory
  - Access to all memory
- 2D-mesh NoC
  - cMesh, on-chip write
  - rMesh, on-chip read
  - xMesh, off-chip write
- No stand alone operation
- Available as 16 core (/64 core) version

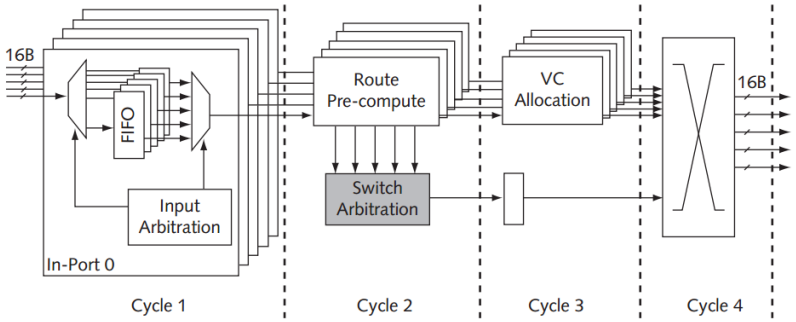


Intel Single Chip Cloud Computer (SCC)



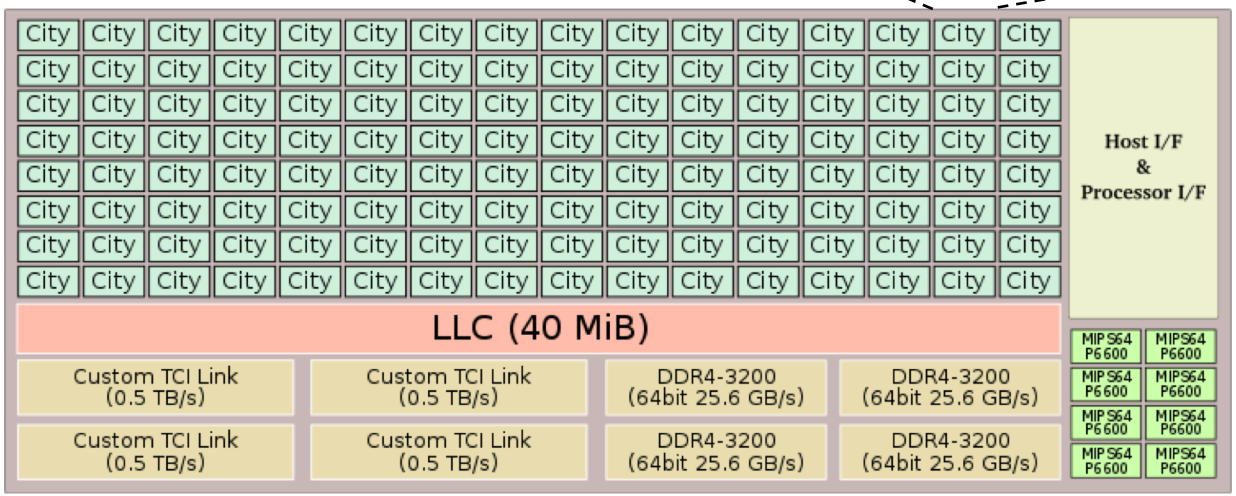
- Intel's research processor
  - 24 tiles
  - 2 cores per tile (P54C-based)
  - →48 cores
- Caches exist on the tiles
- 2D-NoC
  - RR arbitration between the cores of the same tile
  - Packet switched router
  - Credit based flow control
  - Virtual channels

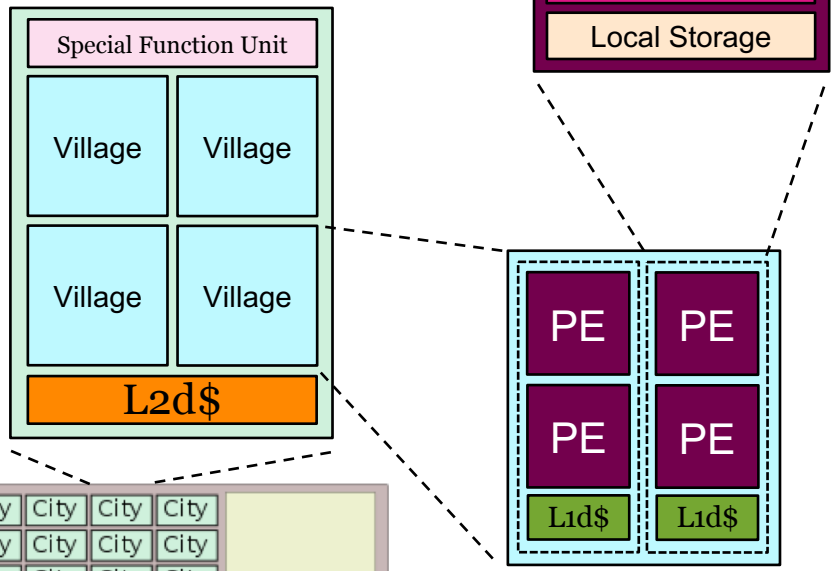


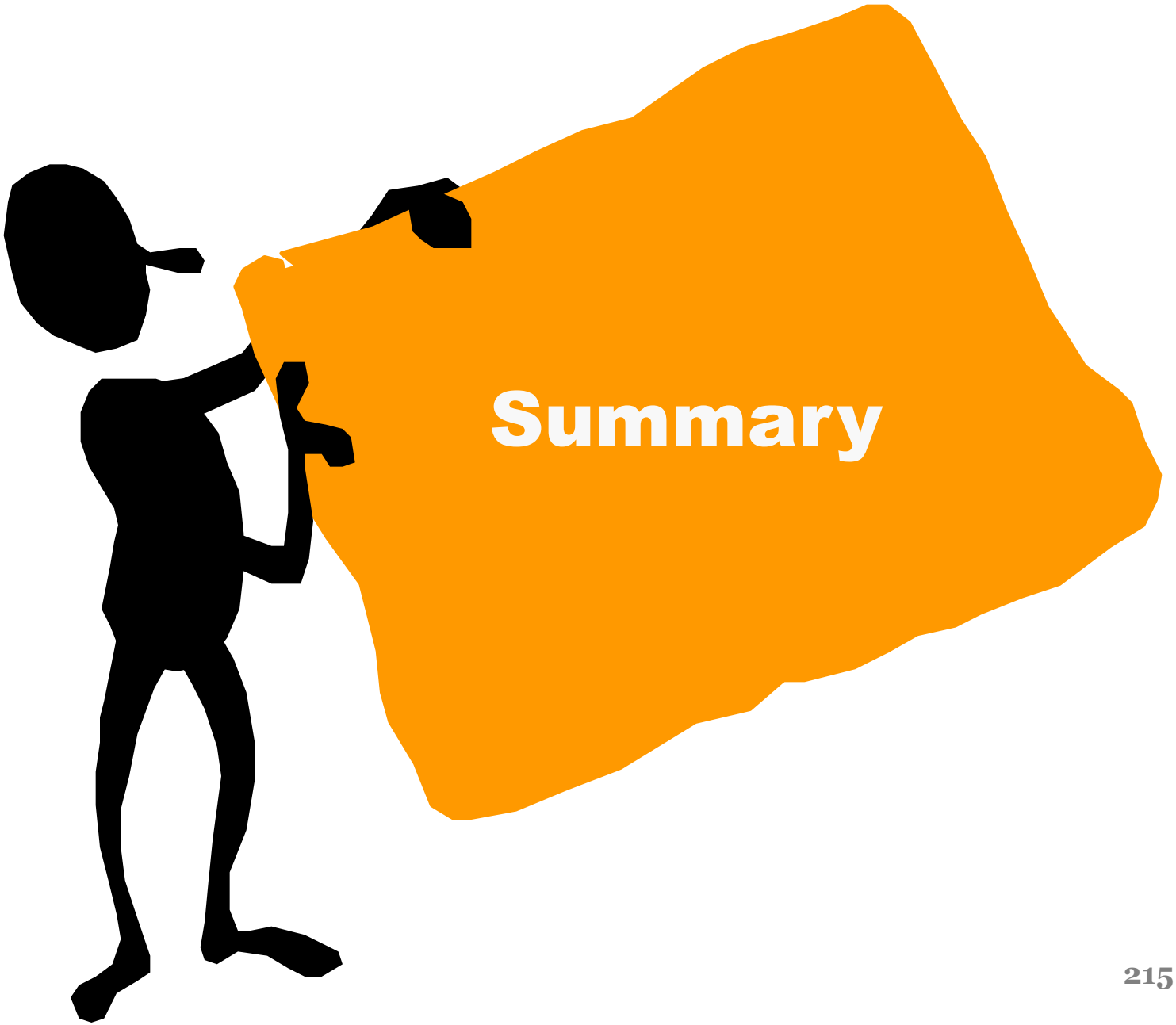


PEZY-SC2 (Super Computer)

- 6x MIPS P6600
- 2048 simple RISC cores
  - 8-way SMT → 16.384 threads
- Shared address space
  - 40MB shared last level cache
- Standalone operation possible



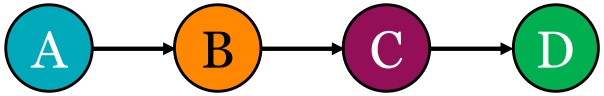



Summary

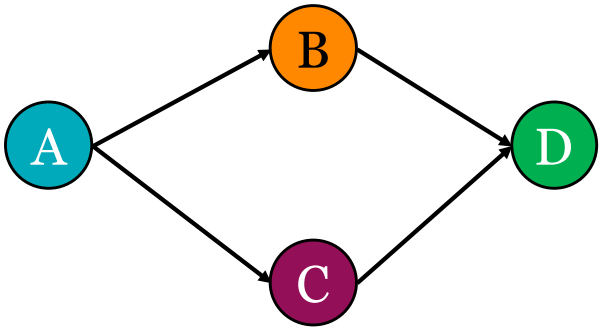
Which workload on which platform?

I

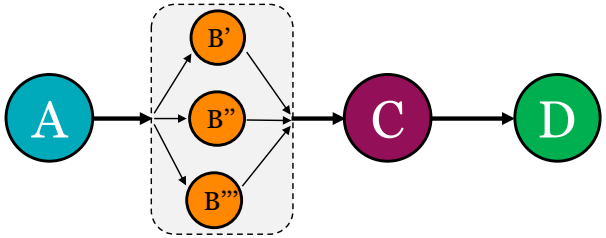
II



III



VI



Summary

This lecture provided an insight into

- Parallel Platforms
  - Heterogeneous Platforms
  - Many-Core Platforms
- Different Workload Characteristics
- Advantages and Challenges of the different Platforms