

JASMIN JAHIC, SIMON BARNER

jj542@cam.ac.uk

barner@fortiss.org

<https://jahic.github.io/hipeac2021>

09:30 - 13:00, 18.01.2021,
BUDAPEST, HUNGARY

HANDLING CONCURRENCY IN EMBEDDED SOFTWARE SYSTEMS FROM ARCHITECTURAL POINT OF VIEW: PART 1

AGENDA

9:30

Session 1: Fundamental Issues with Concurrency in Embedded Software Systems from Architectural Point of View

10:30

10:45

Session 2: Modelling and DSE Methods for Mixed-Critical Software Systems using Multicore Architectures

11:45

12:00

Session 3: Synchronization in Concurrent Software is an Architectural Decision

13:00

AGENDA

9:30

Session 1: Fundamental Issues with Concurrency in Embedded Software Systems from Architectural Point of View

10:30

10:45

Session 2: Modelling and DSE Methods for Mixed-Critical Software Systems using Multicore Architectures

11:45

12:00

Session 3: Synchronization in Concurrent Software is an Architectural Decision

13:00

SESSION 1

9:30

Introduction to the topic

Understand the basics of software system architecture

Understand the basics of computing laws and how they relate to architecture topic

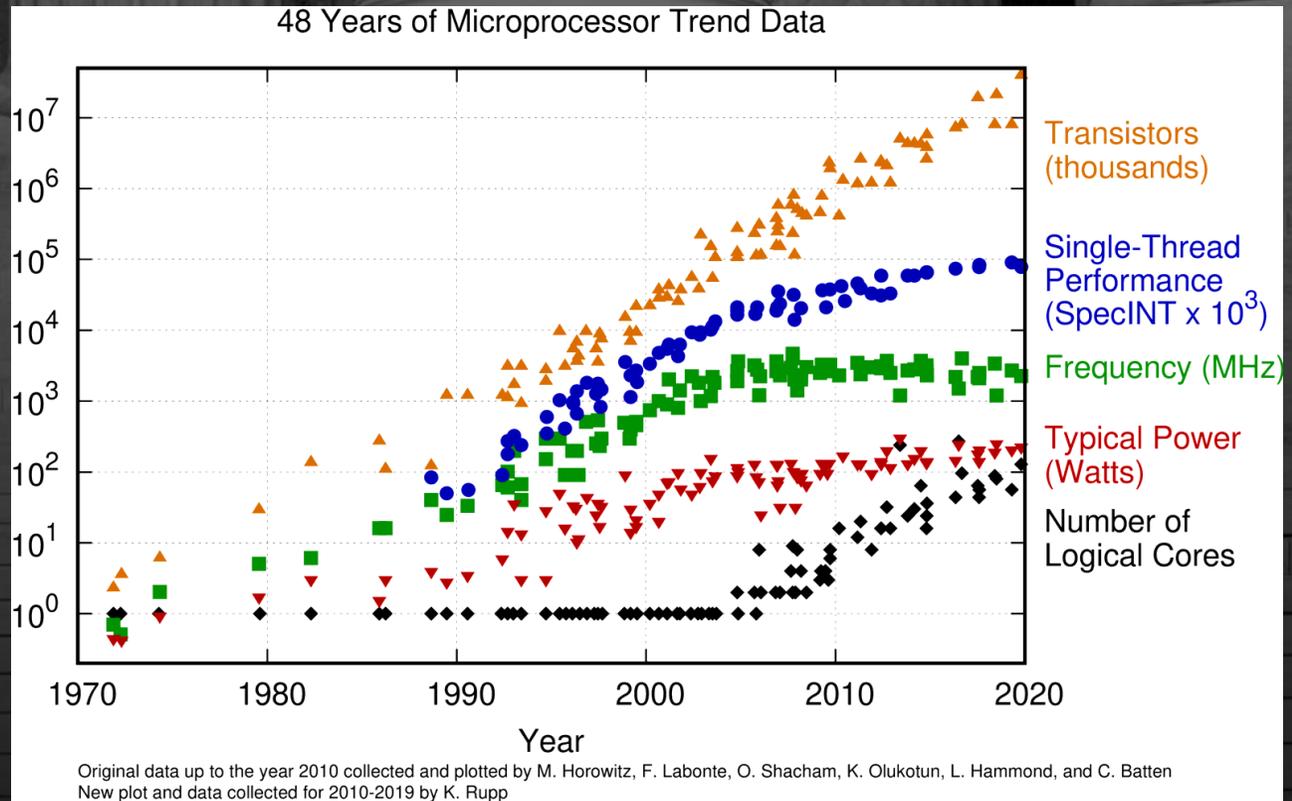
10:30

Understand important architectural properties of embedded systems affected by introducing concurrency

LITERATURE

- [1] The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software, Dr. Dobbs's Journal, 30(3), March 2005
- [2] Software Architecture in Practice, Len Bass, Paul Clements, Rick Kazman, 3rd edition, 2012
- [3] Pragmatic Evaluation of Software Architectures, J. Knodel, M. Naab, 2016
- [4] G. M. Amdahl, "Computer Architecture and Amdahl's Law," in Computer, vol. 46, no. 12, pp. 38-46, Dec. 2013
- [5] A glimpse of real-time systems theory and practice in the wake of multicore processors and mixed-criticality, Tullio Vardanega, University of Padua, Italy, ACACES 2020, HiPEAC - <https://www.hipeac.net/acaces/2020/#/program/courses/8/>
- The Art of Multiprocessor Programming, M. Herlihy, N. Shavit, 2011

MOORE'S LAW AND DENNARD SCALING



<https://github.com/karlrupp/microprocessor-trend-data>

MOORE'S LAW AND DENNARD SCALING



Pentium Dual-
Core, 2007



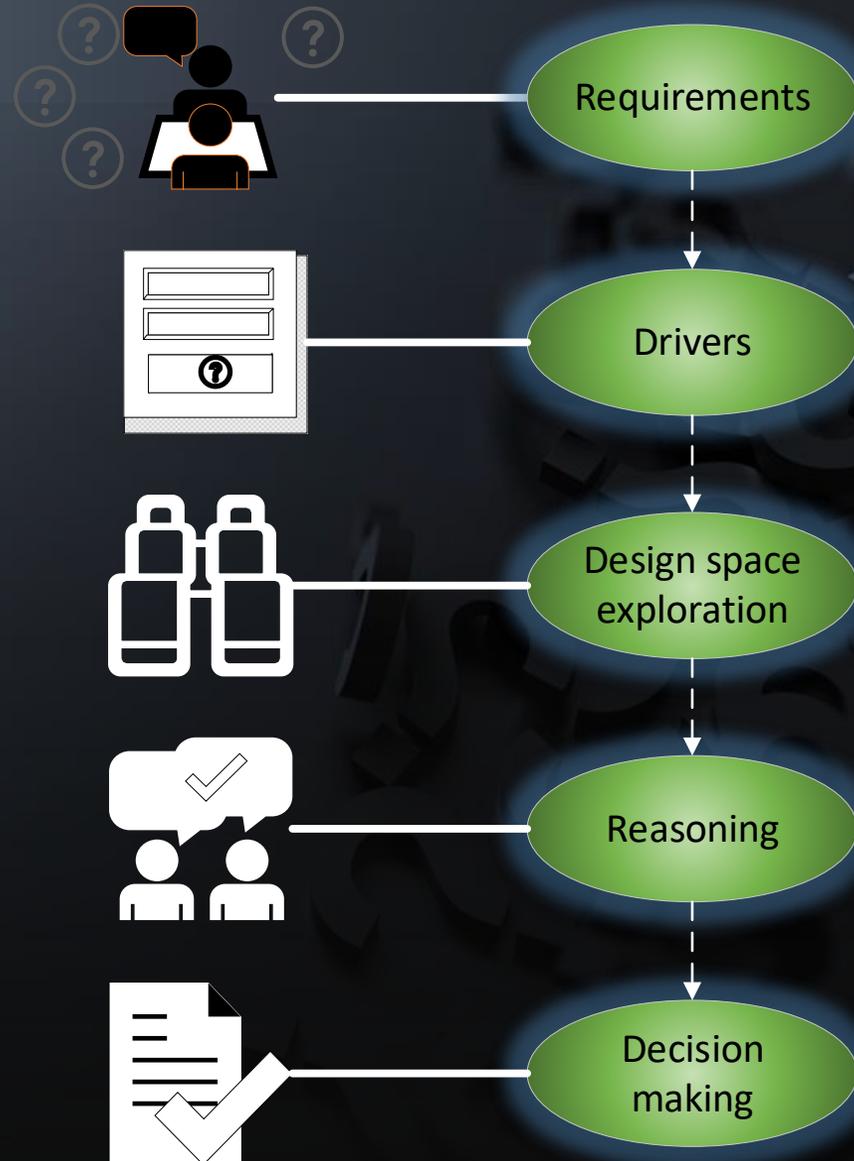
Athlon 64 X2, 2007

- Free lunch: Every new generation of processors would execute with higher frequency – software execution becomes automatically faster – is over! [1]
- Post Dennard scaling breakdown performance drivers:
 - Computer architecture improvements
 - Concurrency and parallelism (forced to use multicores)
 - Power consumption
- Drivers for using multicores
 - Improve execution time
 - Improve throughput
 - Redundancy (availability, reliability)
 - Power consumption
- Without compromising other system quality properties

SOFTWARE SYSTEM ARCHITECTURE

- “Software architecture is the structure of the structures of the system, which comprise software components, the **externally visible properties** of those components, and the **relationships among them.**” [2]
- Requirements
- Drivers
- Decisions

SOFTWARE SYSTEM ARCHITECTURE



SOFTWARE QUALITY

- ISO/IEC 25010:2011 - systems and software quality requirements and evaluation
- ISO/IEC/IEEE 12207 - systems and software engineering - software life cycle processes
- IEEE 730 - software quality assurance
- IEEE 1012 - verification and validation (V&V)

Functional suitability	Performance efficiency	Compatibility	Usability
Functional completeness	Time behaviour	Co-existence	Appropriateness recognizability
Functional correctness	Resource utilization	Interoperability	Learnability
Functional appropriateness	Capacity		Operability
			...

QUALITY DRIVERS

- Quantification of quality in a context
- Quality template [3]

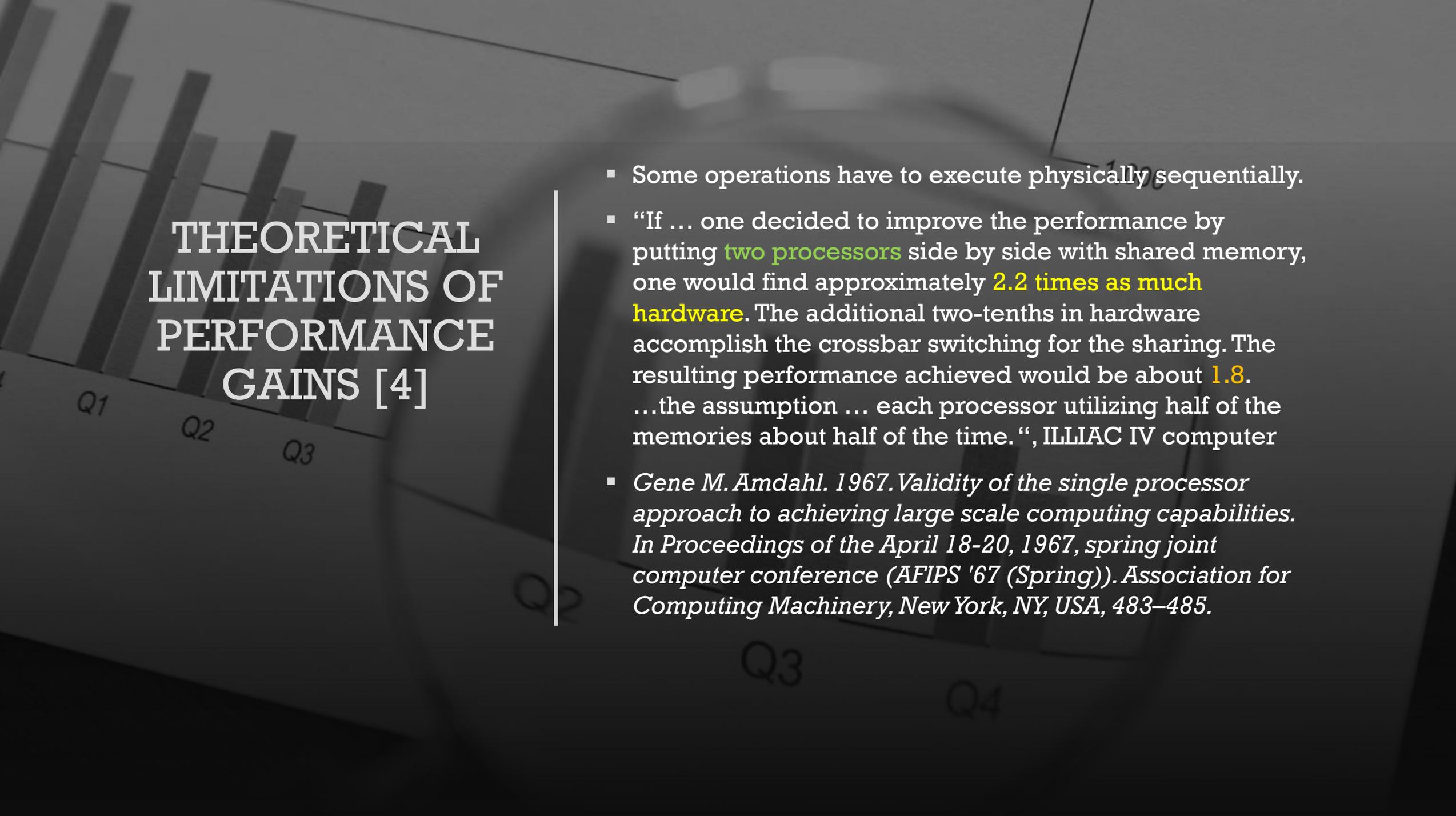
ID	Unique identifier	Status	
Name	Name of scenario	Owner	
Quality	Related quality attribute: exactly one attribute should be chosen.	Stakeholders	
		Quantification	
Environment	Context applying to this scenario. May describe both context and status of the system.		
Stimulus	The event or condition arising from this scenario.		
Response	The expected reaction of the system to the scenario event.		

QUALITY
DRIVERS FOR
ADOPTING
MULTICORES:
SET#1

- Execution time
- Redundancy (availability, reliability)
- Power consumption

EXECUTION TIME:
IDEAL QUALITY
DRIVER
EXPECTATIONS

ID	...	Status	
Name	...	Owner	
Quality	Execution time	Stakeholders	
		Quantification	
Environment	Application software is executing on a single core CPU.	#cores = 1 Execution time = t	
Stimulus	Migrate to a double core CPU	#cores = 2	
Response	Reduce execution time by half.	Execution time = t/2	

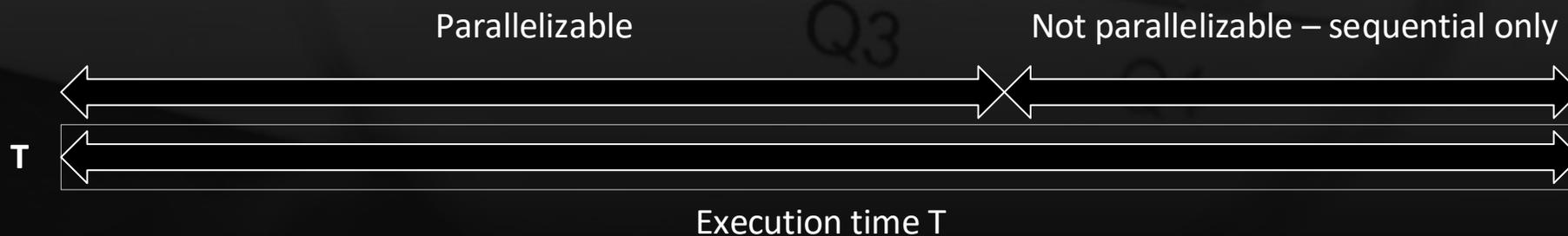


THEORETICAL LIMITATIONS OF PERFORMANCE GAINS [4]

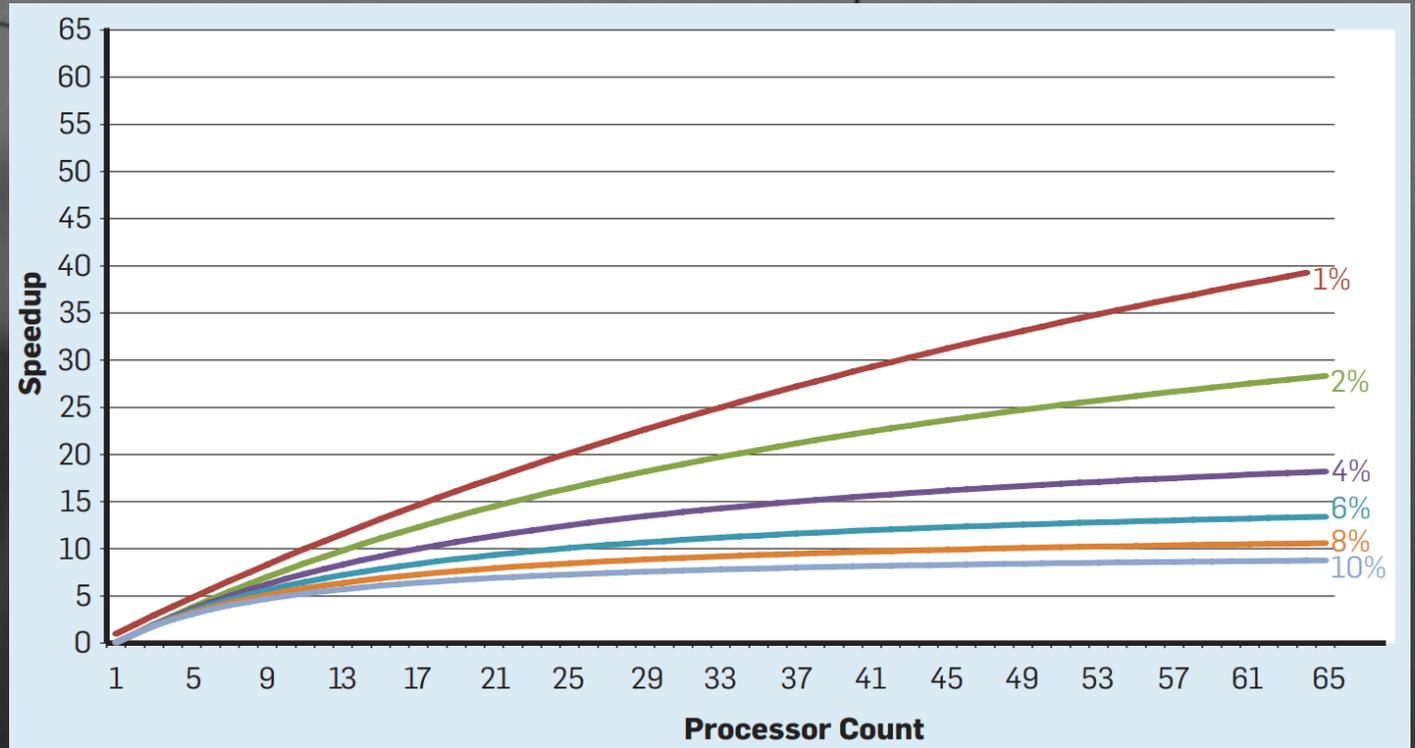
- Some operations have to execute physically sequentially.
- “If ... one decided to improve the performance by putting **two processors** side by side with shared memory, one would find approximately **2.2 times as much hardware**. The additional two-tenths in hardware accomplish the crossbar switching for the sharing. The resulting performance achieved would be about **1.8**. ...the assumption ... each processor utilizing half of the memories about half of the time.”, ILLIAC IV computer
- *Gene M. Amdahl. 1967. Validity of the single processor approach to achieving large scale computing capabilities. In Proceedings of the April 18-20, 1967, spring joint computer conference (AFIPS '67 (Spring)). Association for Computing Machinery, New York, NY, USA, 483–485.*

THEORETICAL LIMITATIONS OF PERFORMANCE GAINS

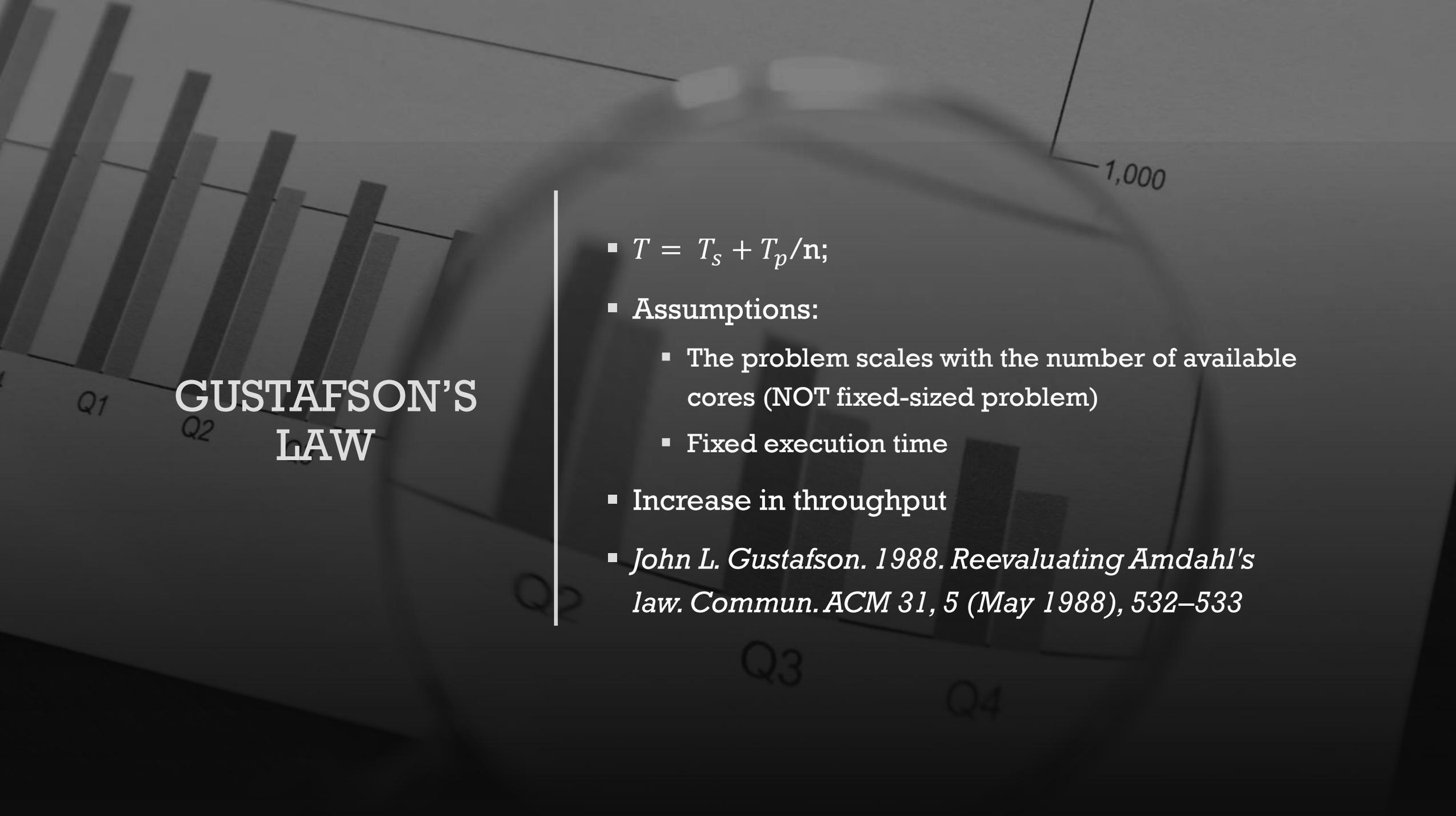
- Some logical problems are hard or impractical to partition into parts that can execute concurrently.
- Amdahl's law
 - $Speedup = \frac{T_s + T_p}{T_s + \frac{T_p}{n}}$; n – number of cores; $T=1$
 - $\frac{1}{T_s + \frac{1-T_s}{n}} \rightarrow (T_s = const.) \rightarrow \lim_{n \rightarrow \infty} \frac{1}{T_s + \frac{T_p}{n}} \simeq \frac{1}{T_s}$
- Assumptions:
 - Fixed-sized problem; T_p is independent of n .
- The slowest task's part limits the speedup



AMDAHL'S LAW



- Effect of Amdahl's law on speedup as a fraction of clock cycle time in serial mode, *John L. Hennessy and David A. Patterson. 2019. A new golden age for computer architecture. Commun. ACM 62, 2 (February 2019), 48–60. DOI:https://doi.org/10.1145/3282307*
- “For example, when only 1% of the time is serial, the speedup for a 64-processor configuration is about 35.”

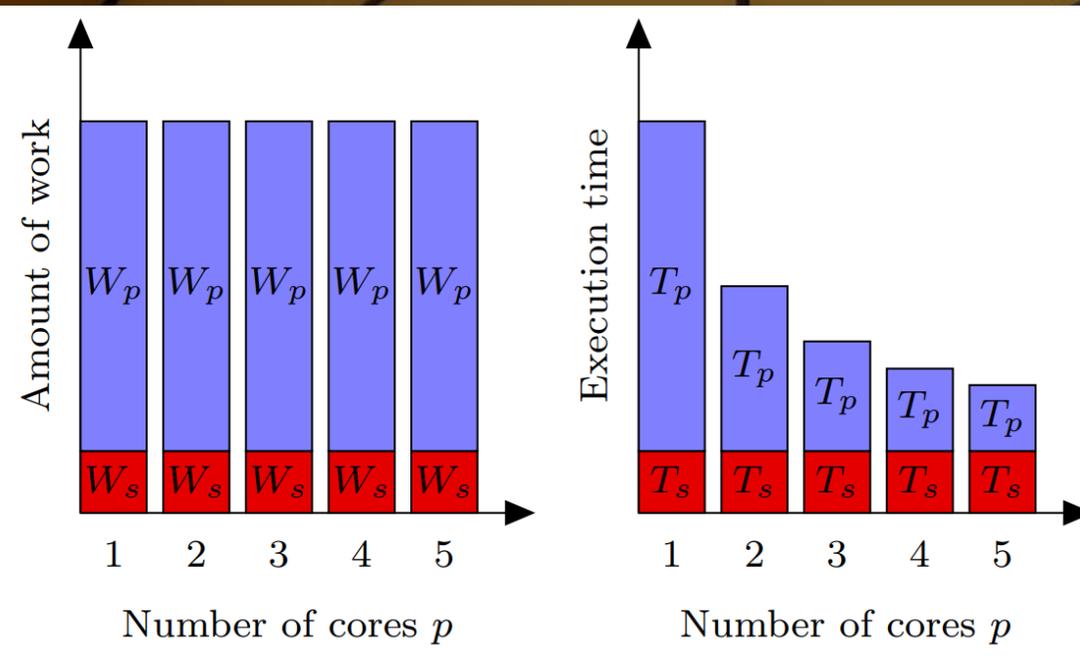


GUSTAFSON'S LAW

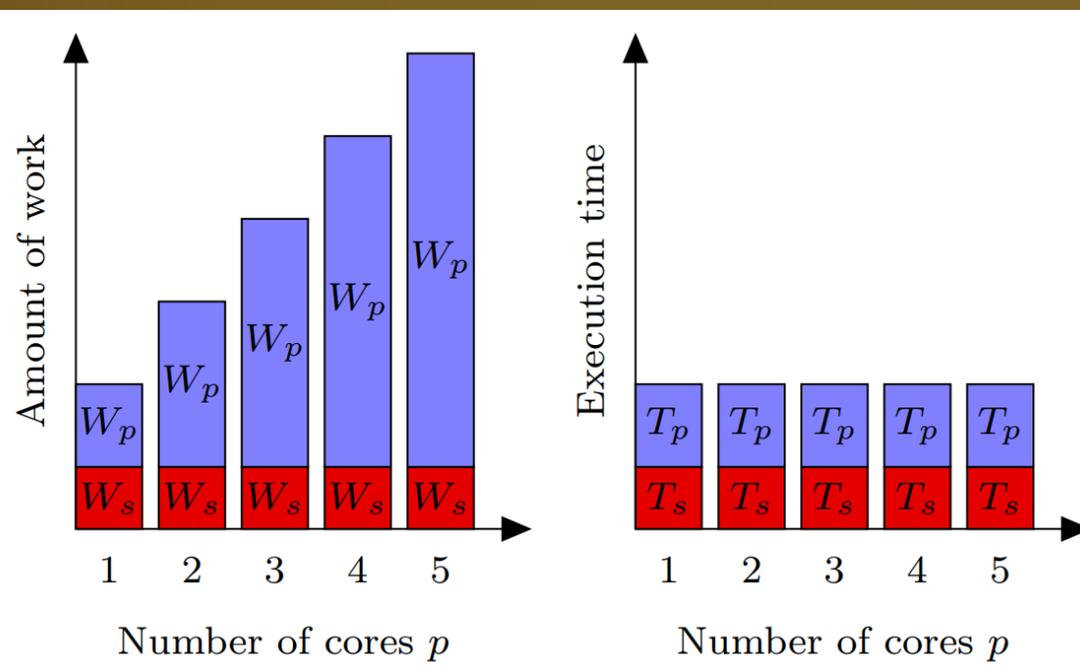
- $T = T_s + T_p/n;$
- Assumptions:
 - The problem scales with the number of available cores (NOT fixed-sized problem)
 - Fixed execution time
- Increase in throughput
- *John L. Gustafson. 1988. Reevaluating Amdahl's law. Commun. ACM 31, 5 (May 1988), 532–533*

AMDAHL'S VS GUSTAFSON ASSUMPTIONS

*B.H.H. Juurlink and C. H. Meenderinck.
2012. Amdahl's law for predicting the
future of multicores considered
harmful. SIGARCH Comput. Archit.
News 40, 2 (May 2012), 1–9.
DOI:[https://doi.org/10.1145/2234336.
2234338](https://doi.org/10.1145/2234336.2234338)*



Amdahl's law

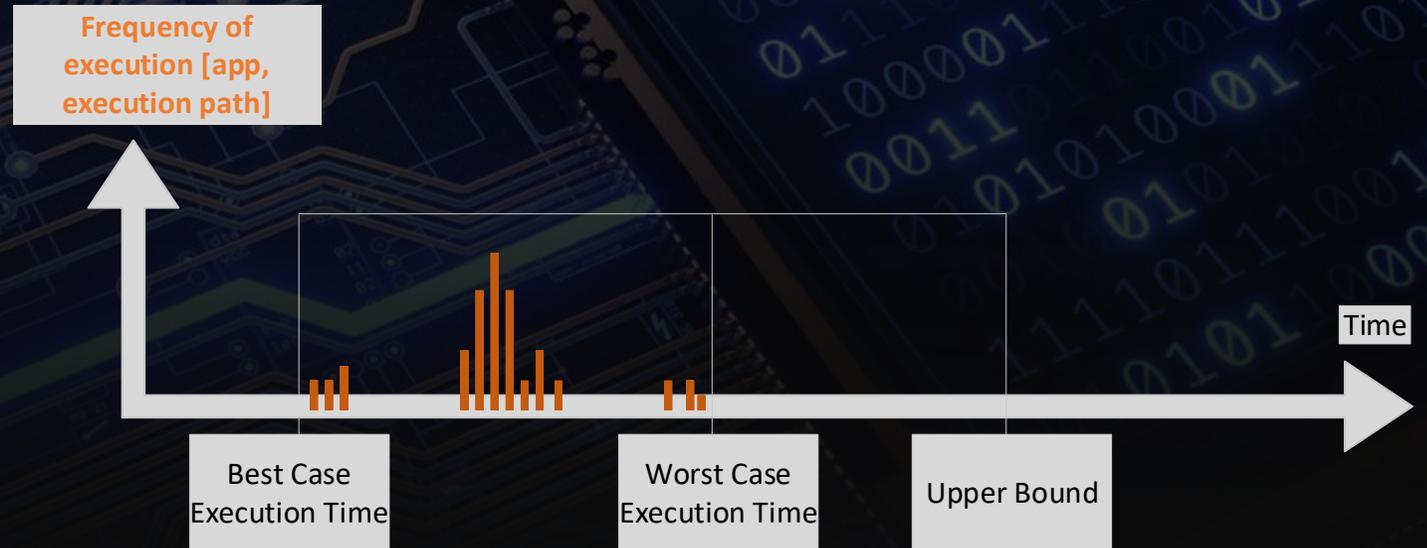


Gustafson's law

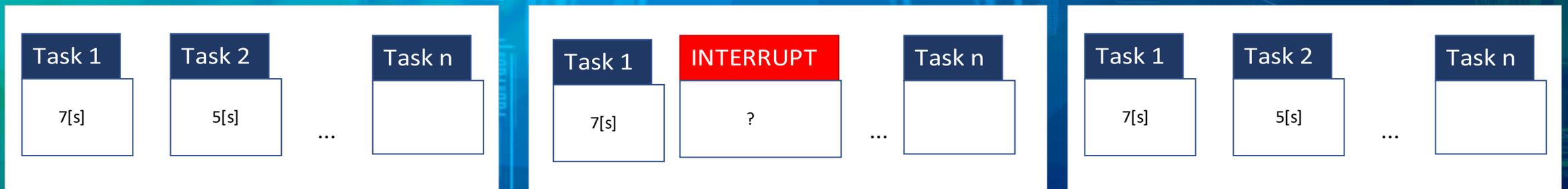
EXECUTION TIME

- Parallelise a single task
- Increase throughput

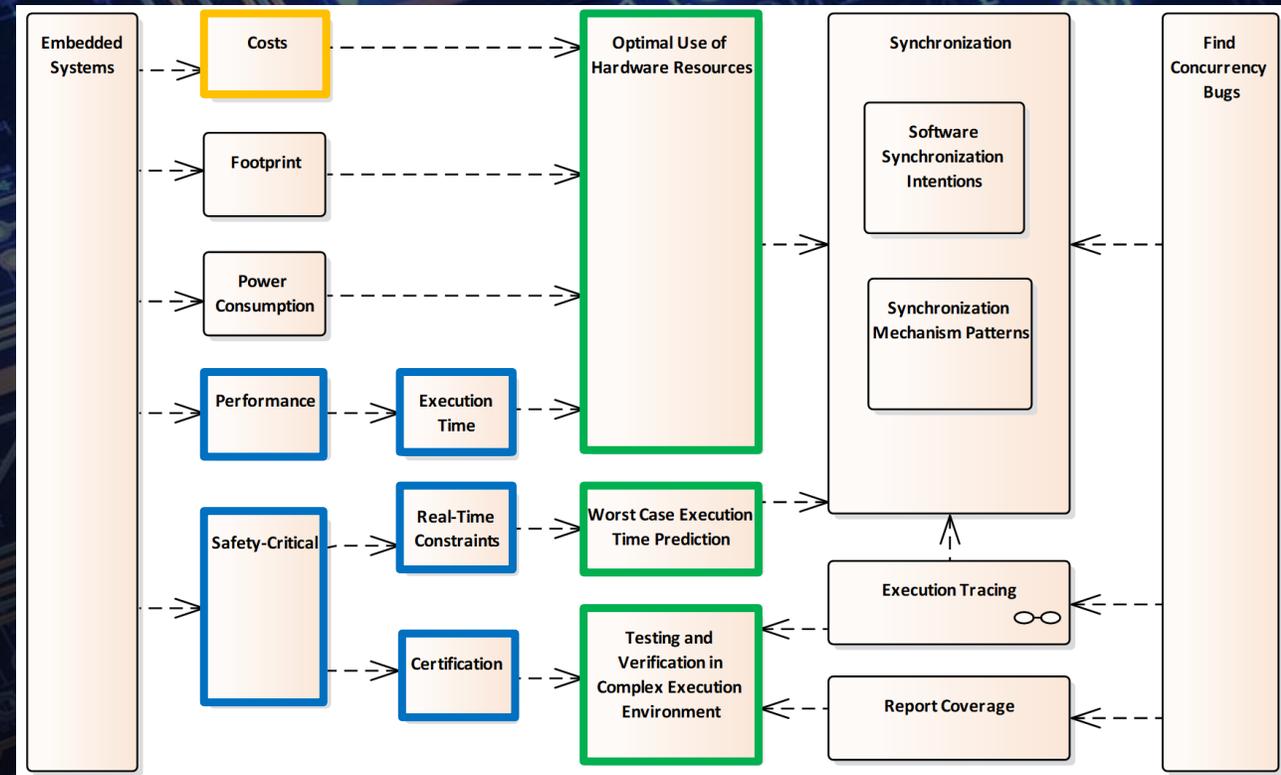
Improve execution time	Average case execution time	Worst case execution time
Single task	User experience	Real-time constraints
Group of tasks	User experience (New features)	Real-time constraints/ Freedom from interference



SOFTWARE IN EMBEDDED SYSTEMS



WHAT COULD POSSIBLY GO WRONG?



Supervised Testing of Embedded Concurrent Software, PhD thesis, Jasmin Jahic, 2020

QUALITY DRIVERS FOR ADOPTING MULTICORES: SET#2

- Average execution time
- User experience
- Real-time constraints
- Safety-critical
- Do not compromise execution correctness

Improve execution time	Average case execution time	Worst case execution time
Single task	User experience	Real-time constraints
Group of tasks	New features	Real-time constraints/ Freedom from interference

QUALITY
PROPERTIES
OF EMBEDDED
SYSTEMS
RELATED TO
MULTICORES



Set#1

Execution time

Redundancy (availability, reliability)

Power consumption

Average execution time

User experience

Set#2

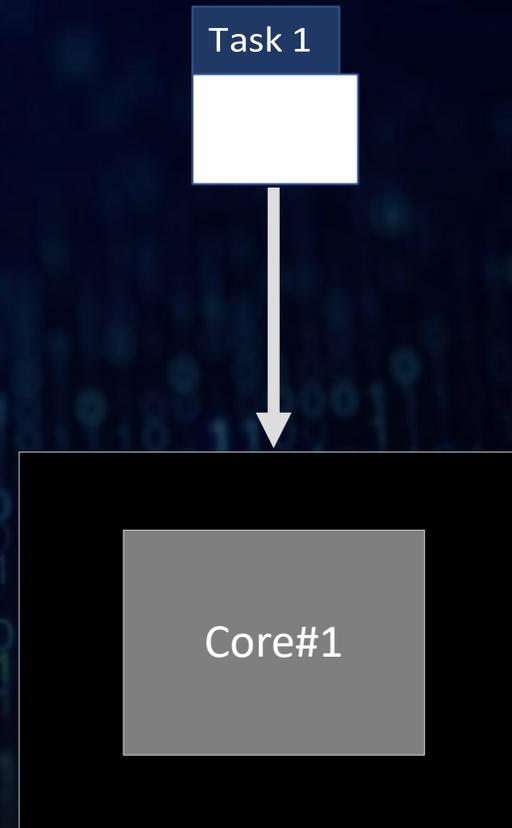
Real-time constraints

Safety-critical

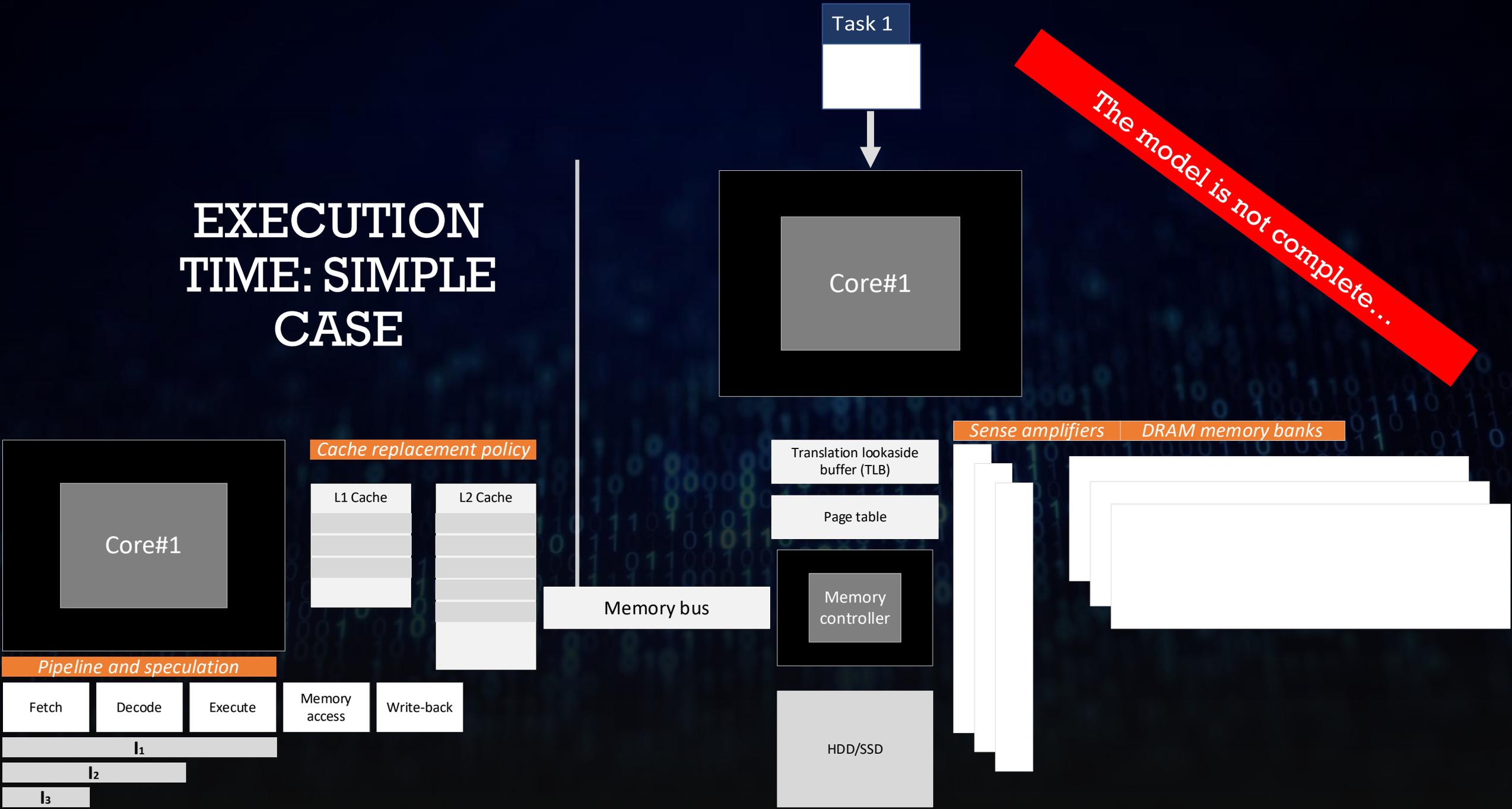
Do not compromise execution correctness



**EXECUTION
TIME: SIMPLE
CASE**



EXECUTION TIME: SIMPLE CASE

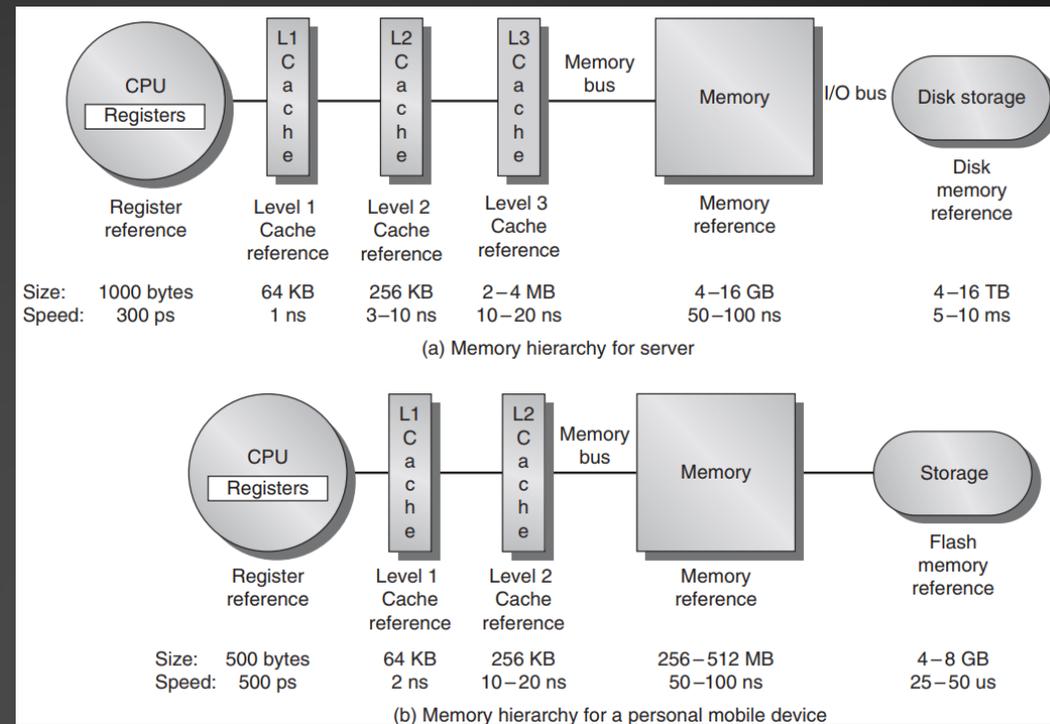


CHALLENGE: EXECUTION TIME

- CPU:
 - Pipelines
 - Speculation
 - Cache behaviour
 - Cache pre-emption
- Memory hierarchy
- ...
- Application software
 - Execution path - Input
- *Design and Analysis of Time-Critical Systems, Jan Reineke, Saarland University, Germany, Summer School ACACES 2017*

Inherently non-deterministic

MEMORY ACCESS

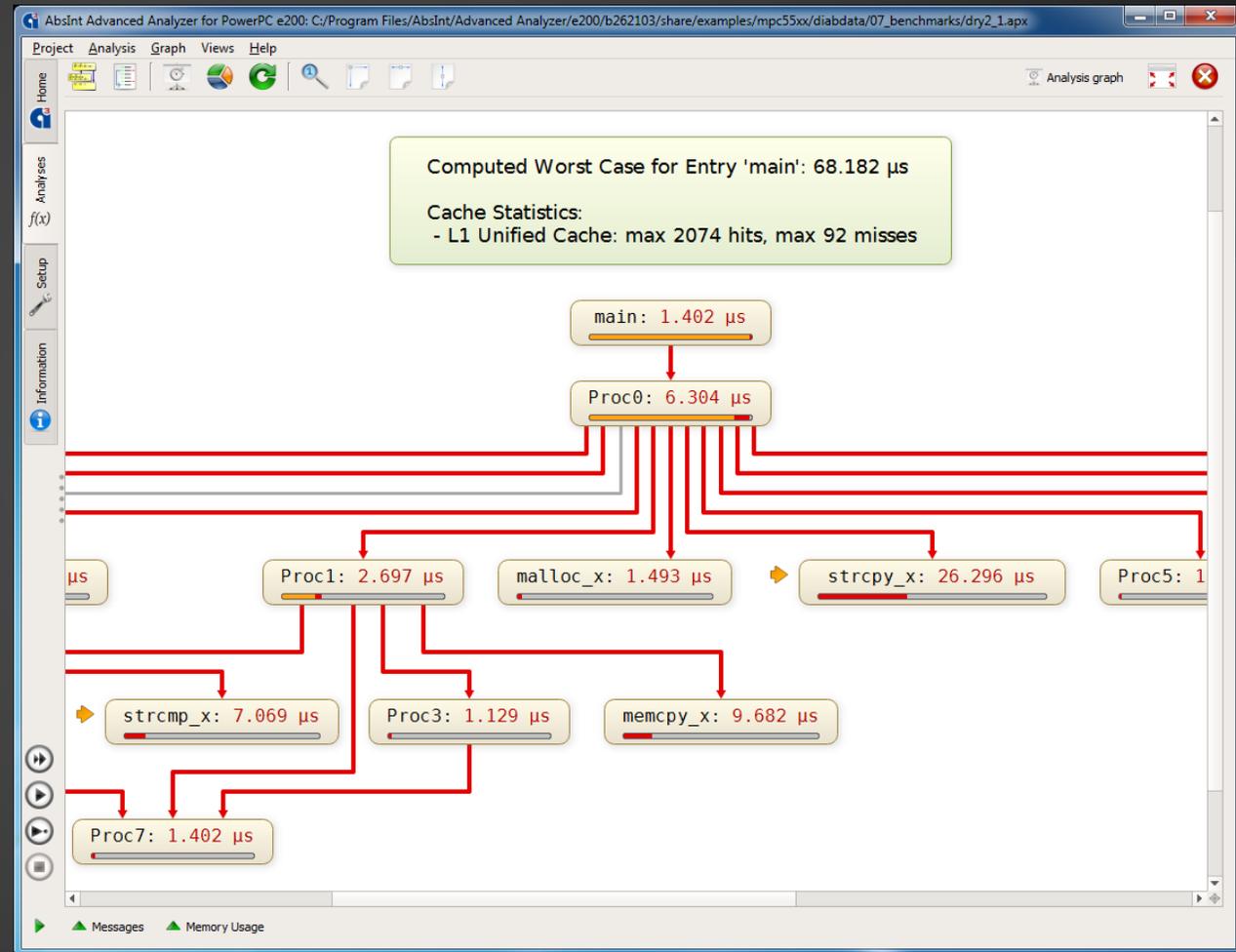


*Computer architecture : a quantitative approach / John L. Hennessy, David A. Patterson.
5th edition, 2011*

Memory technology	Typical access time
SRAM semiconductor memory	0.5–2.5 ns
DRAM semiconductor memory	50–70 ns
Flash semiconductor memory	5,000–50,000 ns
Magnetic disk	5,000,000–20,000,000 ns

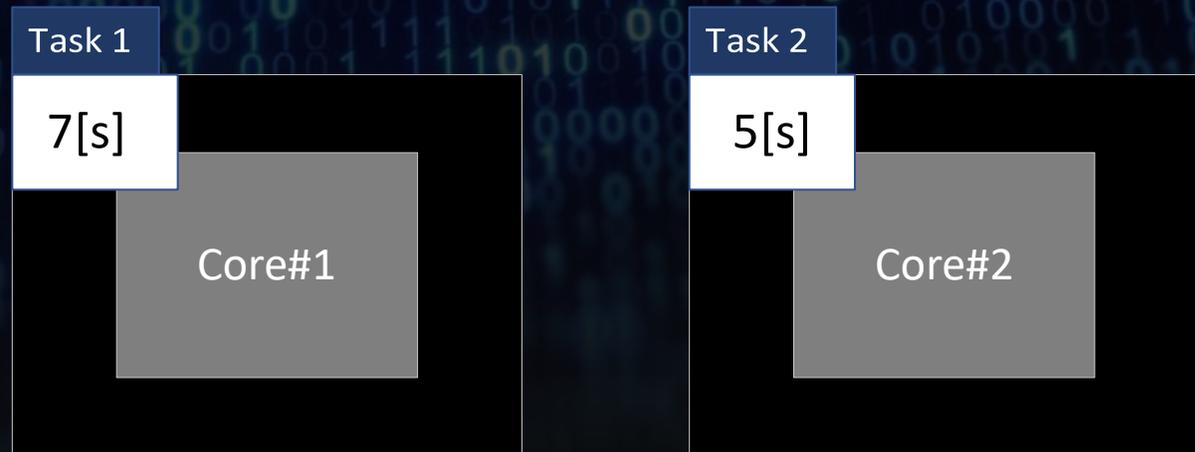
Patterson, D.A. & Hennessy, J.L. (2017). Computer organization and design: The hardware/software interface RISC-V edition

SYSTEM FUNCTIONS



EXECUTION TIME: MULTIPLE TASKS CASE

- Single core execution time: 12 [s]
- Dual-core execution time: 7 [s]
- Speedup: 1.71x



+ Inherently non-deterministic

Cache replacement policy

Cache coherence



Translation lookaside buffer (TLB)

Page table

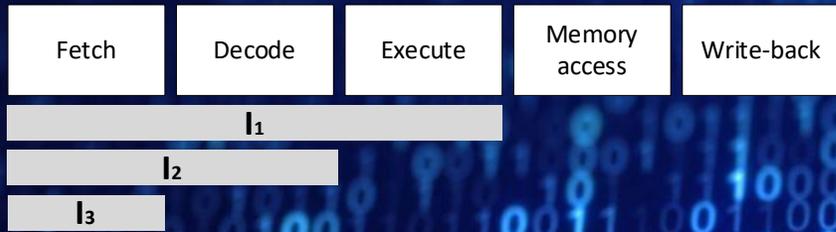
Memory controller

HDD/SSD

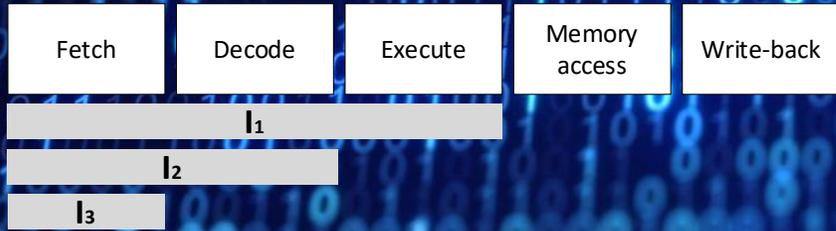
Sense amplifiers | DRAM memory banks



Pipeline and speculation



Pipeline and speculation



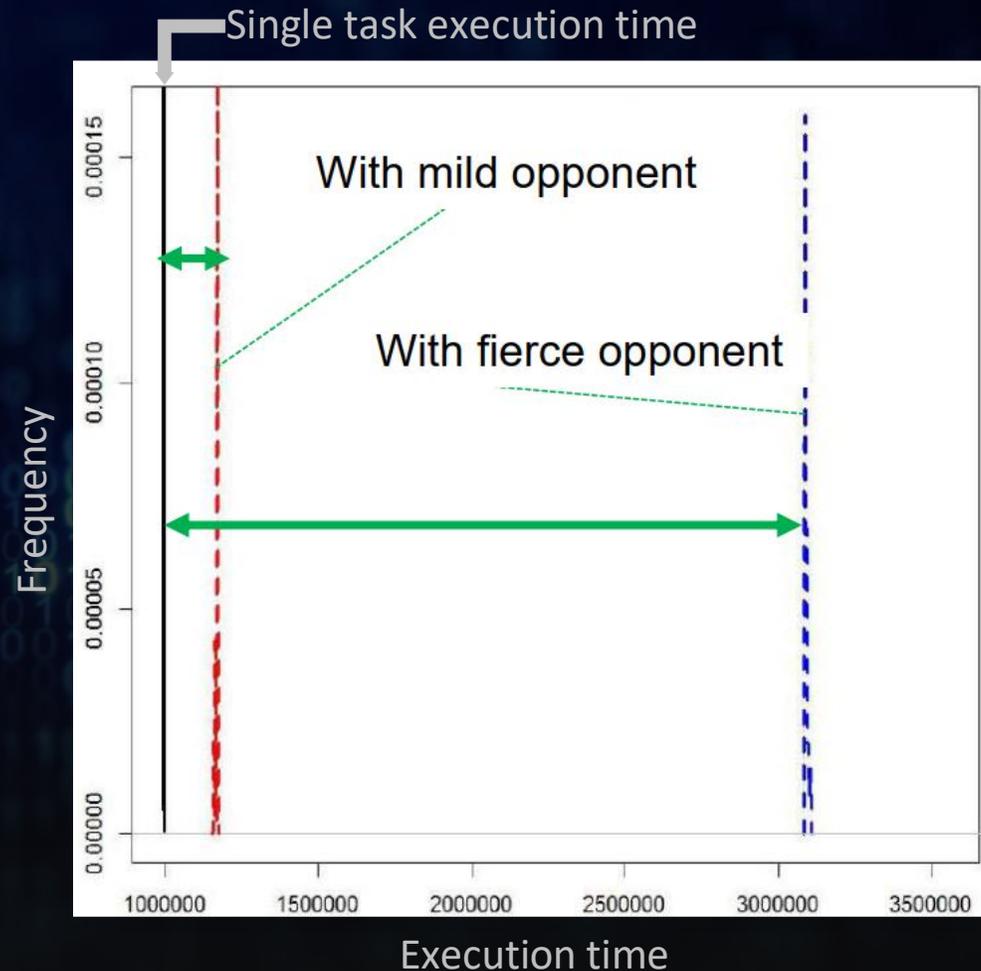
**EXECUTION TIME:
MULTIPLE TASKS CASE**

WCET OF TASKS ON MULTICORES



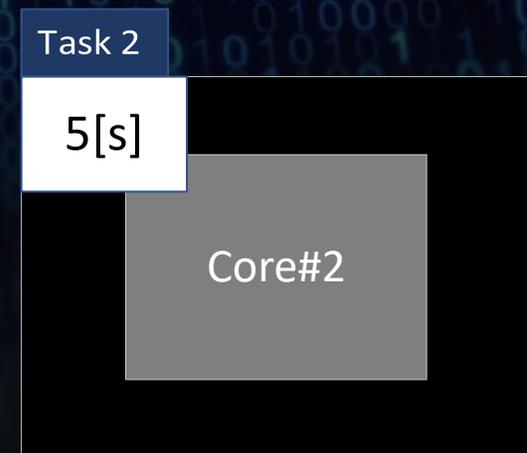
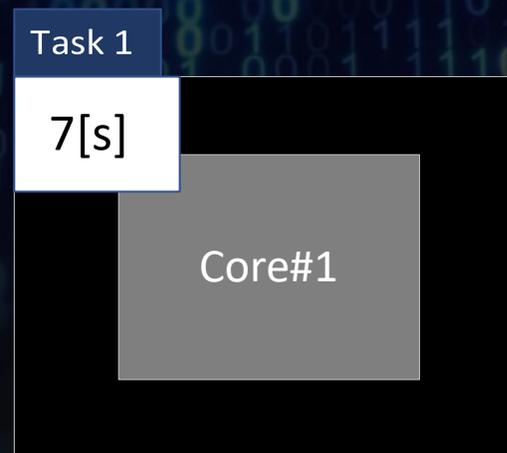
PROARTIS: PRObabilistic Analyzable Real Time Systems - www.rapitasystems.com/about/research-projects/proartis-probabilistic-analyzable-real-time-systems

- “The WCET of even the simplest single-path program running alone on a CPU does not stay the same when other programs run on other CPUs” [5]



EXECUTION TIME: MULTIPLE TASKS CASE

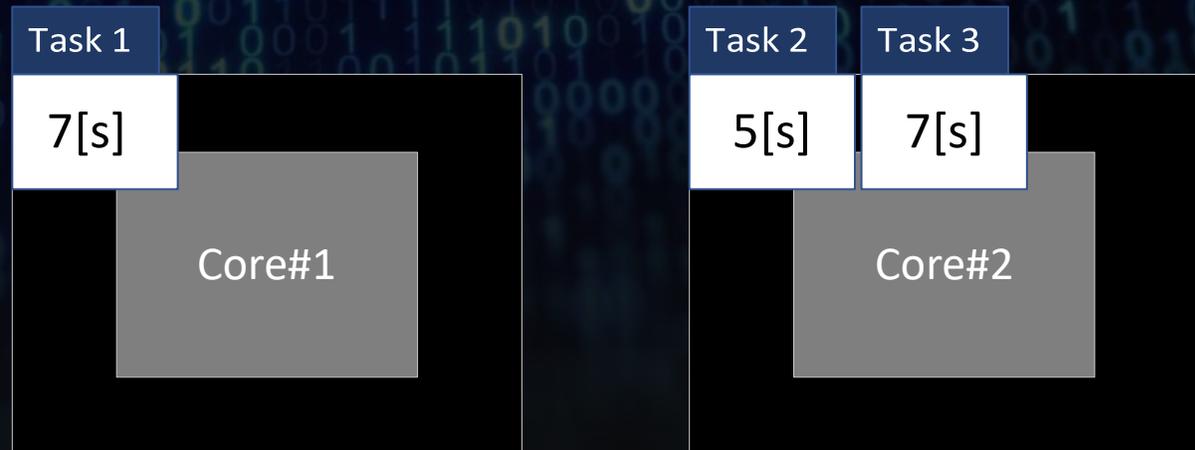
- New task 3: 7 [s]



EXECUTION TIME: MULTIPLE TASKS CASE

- Single core execution time: 19 [s]
- **Dual-core execution time: 12 [s]**
- Speedup: 1.58x

+ Inherently non-deterministic



Tasks scheduling

Cache replacement policy

Cache coherence



Sense amplifiers

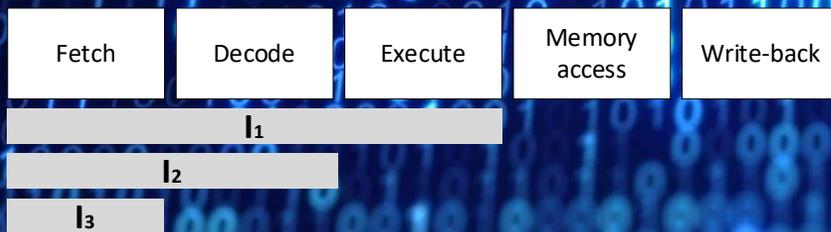
DRAM memory banks



Pipeline and speculation



Pipeline and speculation



**EXECUTION TIME:
MULTIPLE TASKS CASE**

QUALITY
DRIVERS FOR
ADOPTING
MULTICORES:
SET#3

- Core affinity
- Scheduling policy
- Interrupts

SCHEDULING ON MULTICORE PROCESSORS

- Definitions [5]:

- A valid schedule is said to be feasible if it satisfies the temporal constraints of every job.
- A job set is said to be schedulable by a scheduling algorithm if that algorithm always produces a valid schedule for that problem
- A scheduling algorithm is optimal if it always produces a feasible schedule when one exists
- Utilisation U_i of a task T_i : The ratio between execution time (C_i) of a task and a period of time

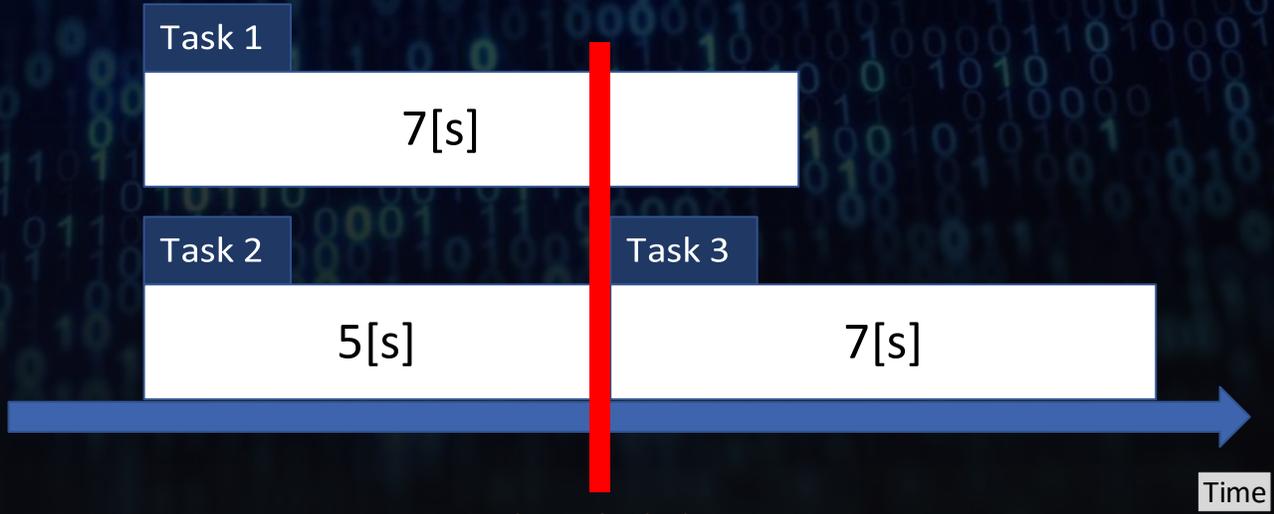
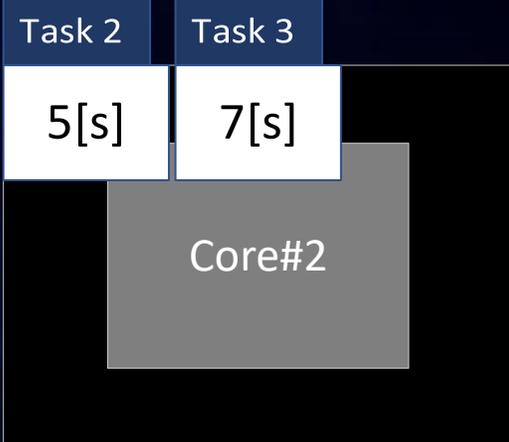
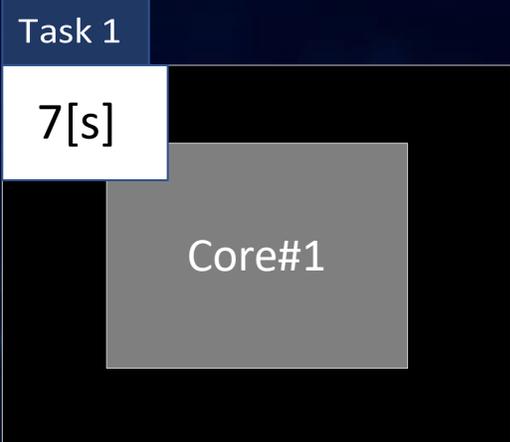
$$P_i: U_i = \frac{C_i}{P_i}$$

- Utilisation for the system: $U = \sum_i U_i < m$; m – number of cores

SCHEDULING ON MULTICORE PROCESSORS

- Utilisation
 - For m resources (cores) and n tasks, how to schedule tasks so to avoid underutilisation of resources? How to avoid idle resources? (without using static scheduling), while at the same time
 - Minimise pre-emption
 - Minimise spinning
- Deadlines
 - **No optimal on-line scheduler can exist for a set of jobs with two or more distinct deadlines on any $(m > 1)$ multiprocessor system.** Theorem [Hong, Leung: RTSS 1988, IEEE TCO 1992]

EXECUTION TIME: MULTIPLE TASKS CASE

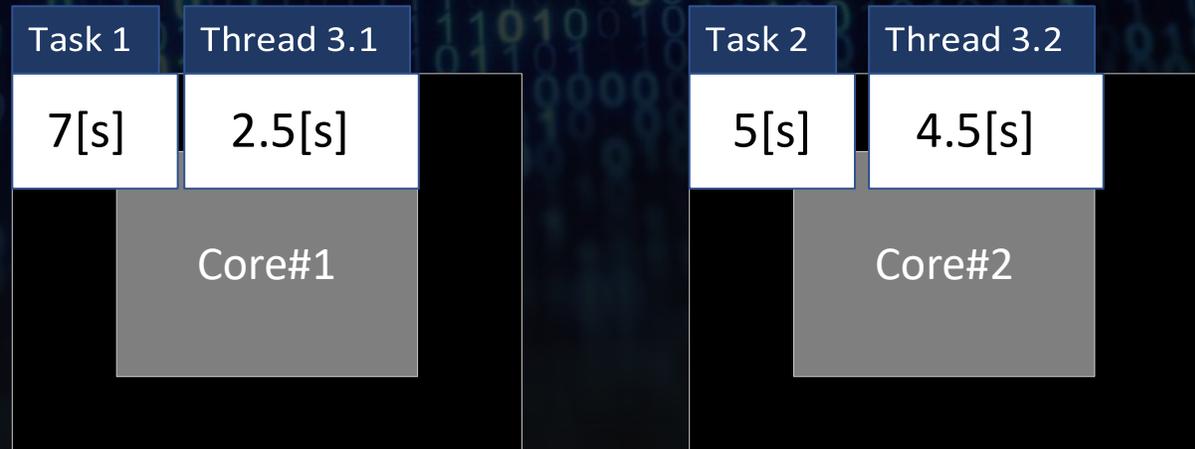


Too late to decide about scheduling...

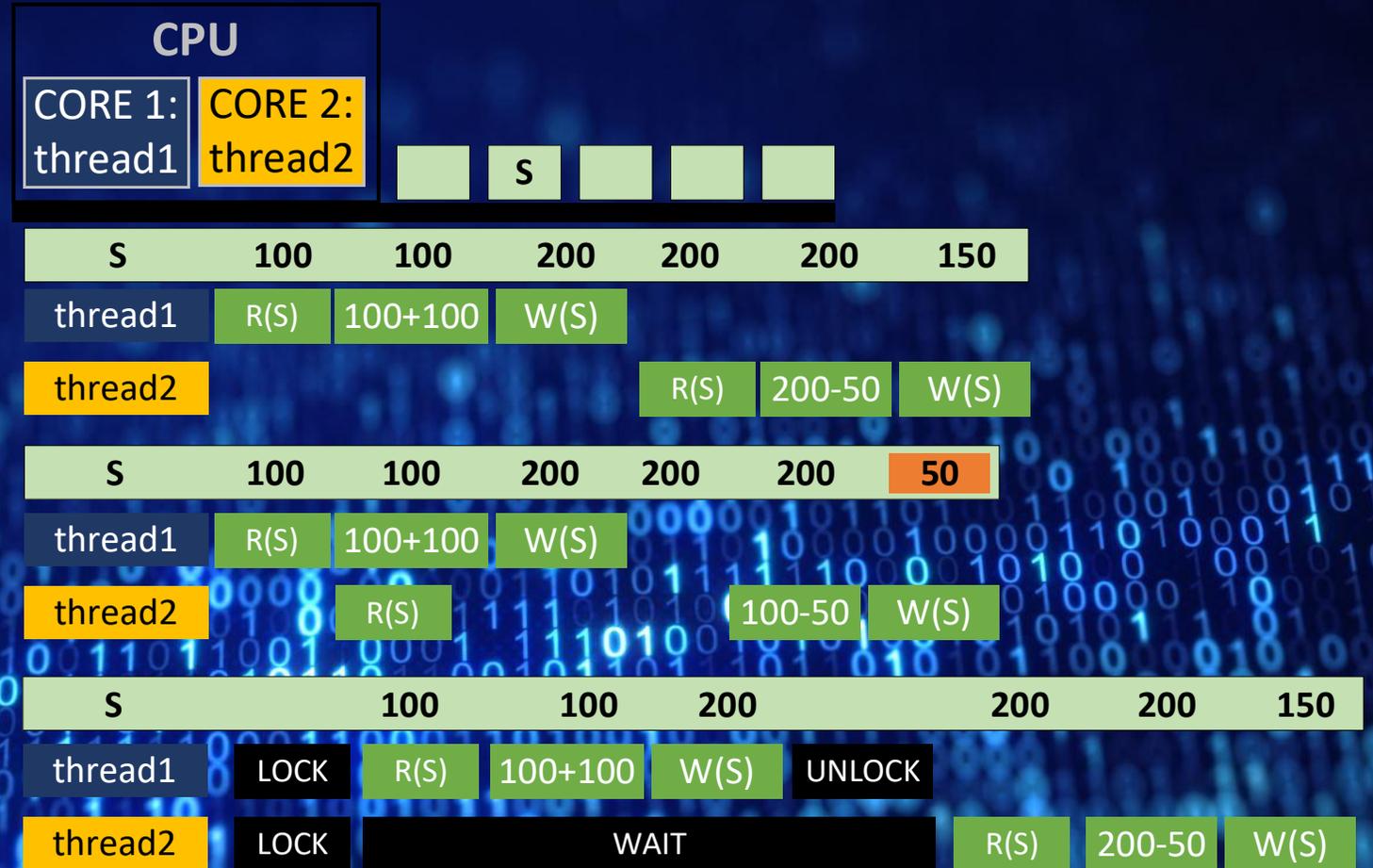
EXECUTION TIME: MULTIPLE THREADS CASE

- Single core execution time: 19 [s]
- Dual-core execution time: 9.5 [s]
- Speedup: **2x (ideally, but not really)**

+ Inherently non-deterministic



CONCURRENCY BUG EXAMPLE



QUALITY DRIVERS FOR ADOPTING MULTICORES: SET#4

- Ways and means to partition software - partitioning strategy
- Thread start-up time
- Synchronisation
- Liveness
- Concurrency bugs
- Bugs that exist on execution paths possible only because of concurrency

QUALITY PROPERTIES OF EMBEDDED SYSTEMS RELATED TO MULTICORES

Set#1

Execution time

Redundancy
(availability, reliability)
Power consumption

Set#2

Average execution
time

User experience

Real-time constraints

Safety-critical

Do not compromise
execution correctness

Set#3

Core affinity

Scheduling policy

Interrupts

Set#4

Ways and means to
partition software -
partitioning strategy

Thread start-up time

Synchronisation

Liveness

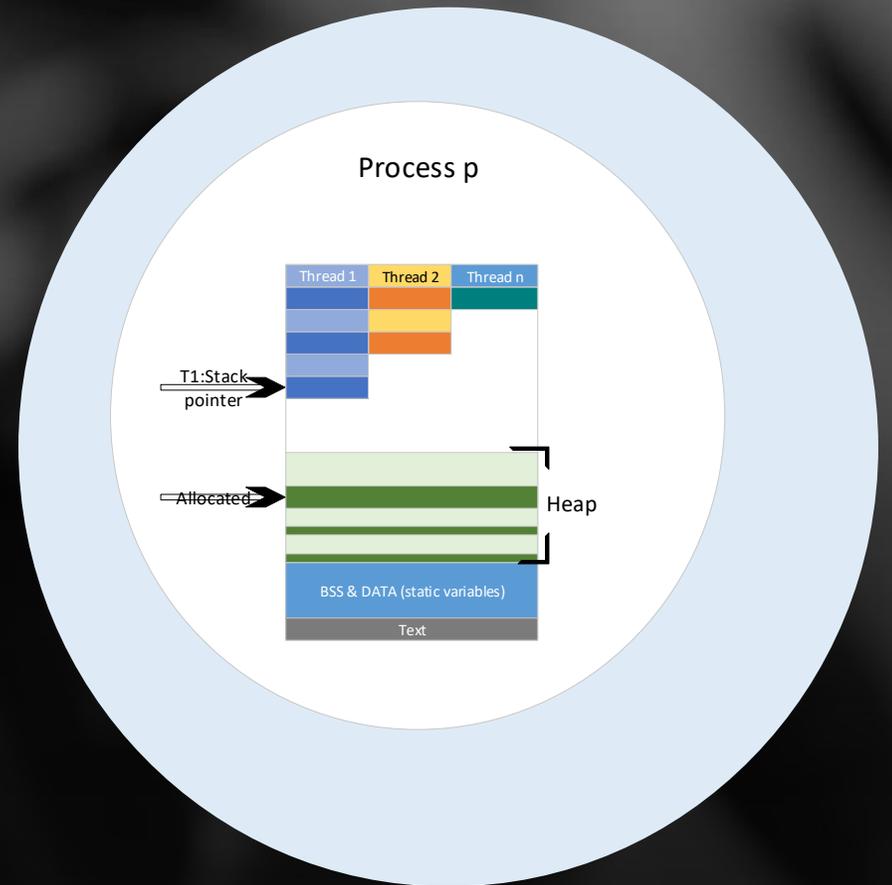
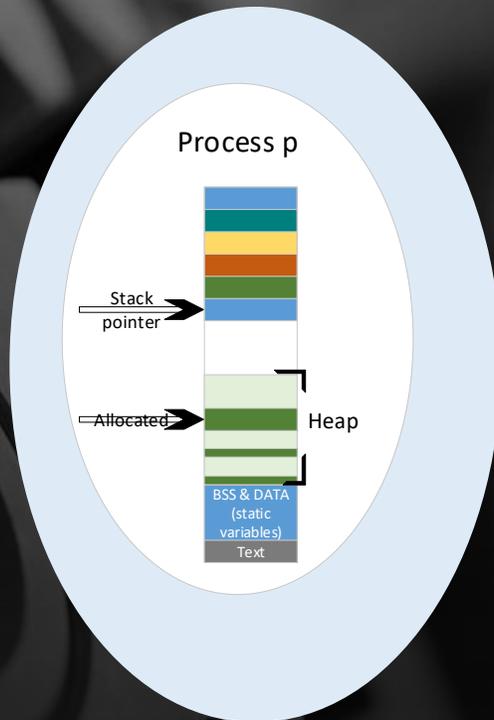
Concurrency bugs

Bugs that exist on
execution paths
possible only because
of concurrency

COMPUTER ARCHITECTURE IMPROVEMENTS

- CPU performance (time): $\frac{\text{Instruction count} * \text{CPI}}{\text{Clock rate}}$
 - Instruction count
 - CPI - cycles per instruction
 - Clock rate
- Focus on architectural improvements and how to use the larger number of transistors without being reliant on silicon performance improvements
- Instruction set (e.g., RISC-V)
- Instruction-level parallelism - Pipelining
- Data-level parallelism
- Prediction (e.g., branch prediction)

A MULTITHREADED PROCESS

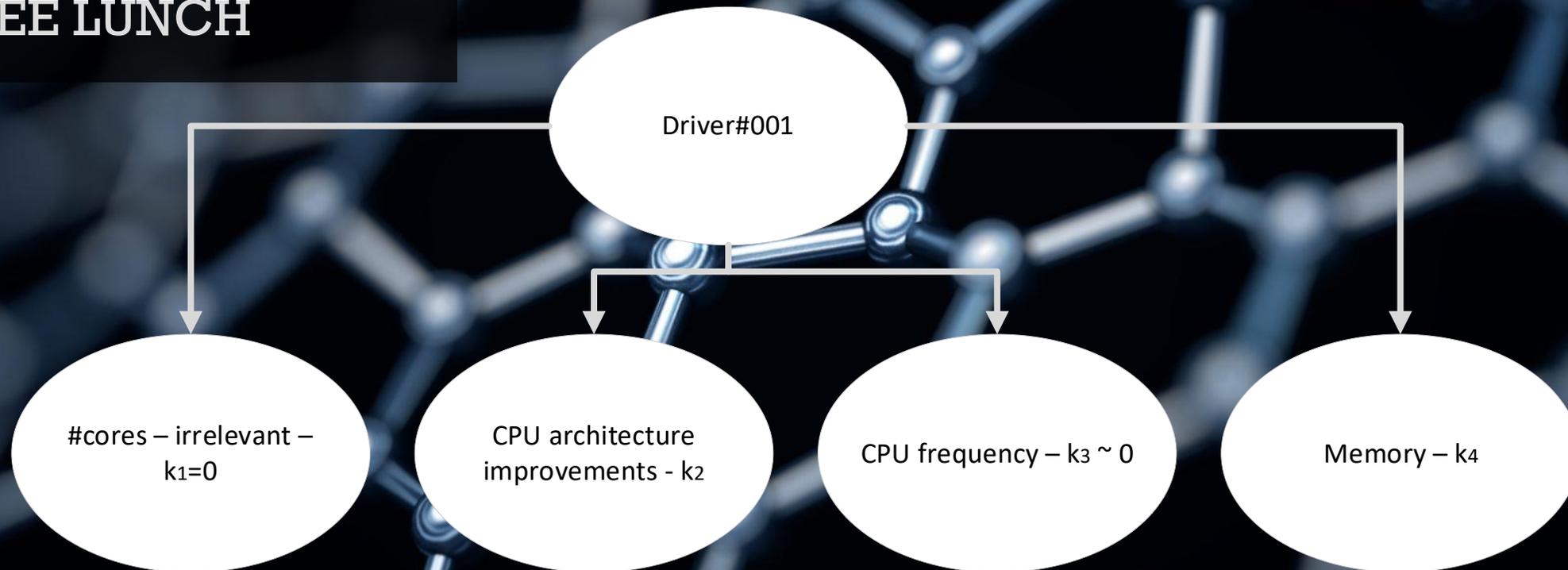


„...each thread runs independently of the others, and each thread may run a different sequence of instructions.“, C++ Concurrency in action, practical multithreading, Anthony Williams, 2012

FREE LUNCH

ID	001	Status	
Name	...	Owner	
Quality	Average case execution time – single task – no partitioning	Stakeholders	
		Quantification	
Environment	Single task is executing on a CPU	Execution time = t	
Stimulus	Migrate to a new hardware (CPU) generation platform	#cores, CPU architecture improvements, CPU frequency, memory (size, speed, hierarchy)	
Response	Significantly reduced (by factor k) execution time	Execution time = t/k	

FREE LUNCH



Execution time = t/k
 $k = k_1 + k_2 + k_3 + k_4$

FREE LUNCH

ID	001	Status	
Name	...	Owner	
Quality	Average case execution time – single task – no new tasks - no partitioning	Stakeholders	
		Quantification	
Environment	Single task is executing on a CPU	Execution time = t	
Stimulus	Migrate to a new hardware (CPU) generation platform	#cores, CPU architecture improvements, CPU frequency, memory (size, speed, hierarchy)	
Response	Significantly reduced (by factor k) execution time	Execution time = t/k	

THROUGHPUT AND USER EXPERIENCE

Set#3

Core affinity

Scheduling policy

Interrupts

ID	002	Status	
Name	...	Owner	
Quality	Average case execution time – multiple tasks – no new tasks - no partitioning	Stakeholders	
		Quantification	
Environment	Multiple tasks are executing on a CPU	Execution time = t	
Stimulus	Migrate to a new hardware (CPU) generation platform	#cores, CPU architecture improvements, CPU frequency, memory (size, speed, hierarchy), set#3 params	
Response	Significantly reduced (by factor k) execution time	Execution time = t/k	

THROUGHPUT AND NEW FEATURES

Set#3

Core affinity

Scheduling policy

Interrupts

ID	003	Status	
Name	...	Owner	
Quality	Average case execution time – multiple tasks – new tasks – no partitioning	Stakeholders	
		Quantification	
Environment	Multiple tasks are executing on a CPU	Execution time = t	
Stimulus	Add new features/new tasks and reconfigure the system	#features (and their requirements), set#3 params	
Response	System runs with the new features, and with a new execution time that is acceptable	#newFeatures, new execution time	

THROUGHPUT AND RE-CONFIGURATION

Set#3

Core affinity

Scheduling policy

Interrupts

ID	004	Status	
Name	...	Owner	
Quality	Average case execution time – multiple tasks – no new tasks - no partitioning	Stakeholders	
		Quantification	
Environment	Multiple tasks are executing on a multicore CPU	Execution time = t	
Stimulus	Configure set#3 parameters	set#3 params	
Response	Significantly reduced (by factor k) execution time	Execution time = t/k	

SPEEDUP OF A SINGLE TASK

Set#3

Core affinity
 Scheduling policy
 Interrupts

Set#4

Ways and means to partition software -
 partitioning strategy

Thread start-up time

Synchronisation

Liveness

Concurrency bugs

Bugs that exist on execution paths possible
 only because of concurrency

ID	005	Status	
Name	...	Owner	
Quality	Average case execution time – single task – partitioning – no dependencies	Stakeholders	
		Quantification	
Environment	Task is executing on a CPU	Execution time = t ; #cores > 1	
Stimulus	Partition the task into threads	#threads > 1, set#3 params, set#4 params (partitioning strategy , thread start-up time)	
Response	Significantly reduced (by factor k) execution time	Execution time = t/k	

SPEEDUP OF A SINGLE TASK

Set#3

Core affinity
Scheduling policy
Interrupts

Set#4

Ways and means to partition software - partitioning strategy
Thread start-up time
Synchronisation
Liveness
Concurrency bugs
Bugs that exist on execution paths possible only because of concurrency

ID	006	Status	
Name	...	Owner	
Quality	Average case execution time – single task – partitioning – dependencies, shared memory	Stakeholders	
		Quantification	
Environment	Task is executing on a CPU	Execution time = t ; #cores > 1	
Stimulus	Partition the task into threads	#threads > 1, set#4 params, set#3 params	
Response	Significantly reduced (by factor k) execution time	Execution time = t/k	

SOFTWARE PARTITIONING - MULTITHREADING

- What else is affected by partitioning software tasks into threads?
- Part 3: Synchronization in Concurrent Software is an Architectural Decision

WHAT ABOUT WORST CASE EXECUTION TIME?

Set#3

Core affinity

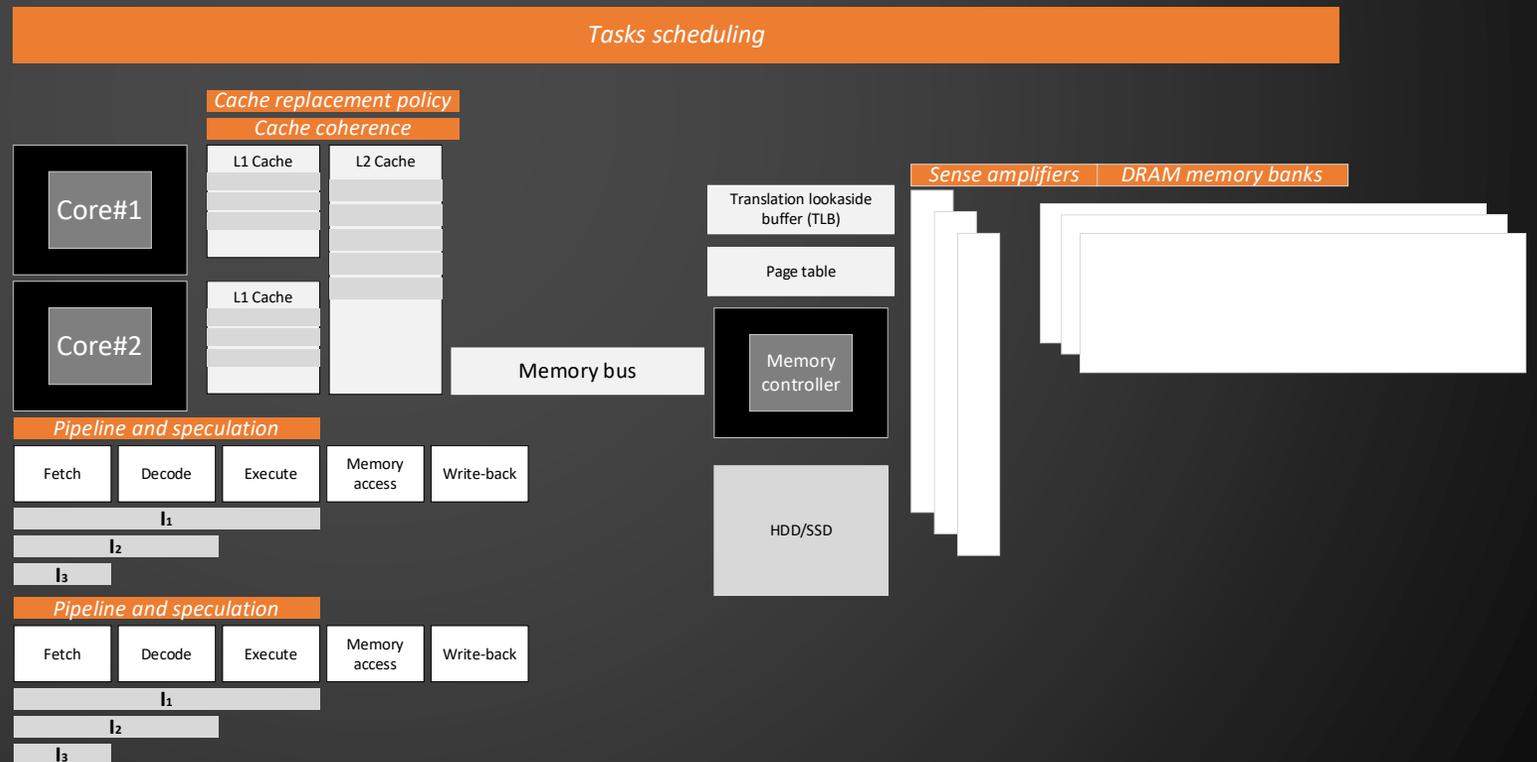
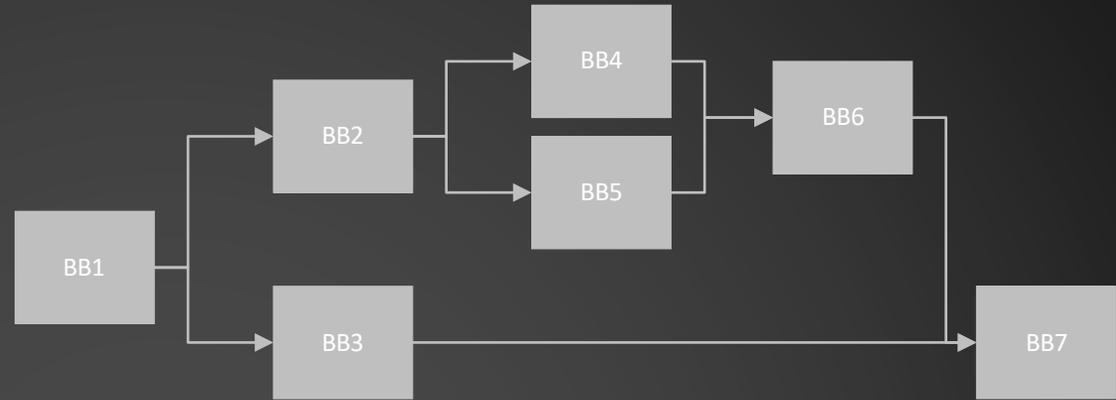
Scheduling policy

Interrupts

- We can try and limit concurrency (set#3 parameters)
- In general, more cores and more tasks makes it harder to predict WCET – increase hardware interference
- Optimal scheduling in multicores
 - Some theoretical concepts – hard to implement [5] (RTOS not ready)
- Use multicores to decrease WCET?
 - Not (always) a good idea [5]

SOME APPROACHES FOR PREDICTING EXECUTION TIME

- Usually WCET
- Precision Timed (PRET) Machines - ptolemy.berkeley.edu/projects/chess/pret/
- aiT WCET Analyzers - www.absint.com/ait
 - Binary executables
 - Intrinsic cache and pipeline behavior
- Timing Behavior of AUTOSAR Multi-Core ECUs - www.timing-architects.com/

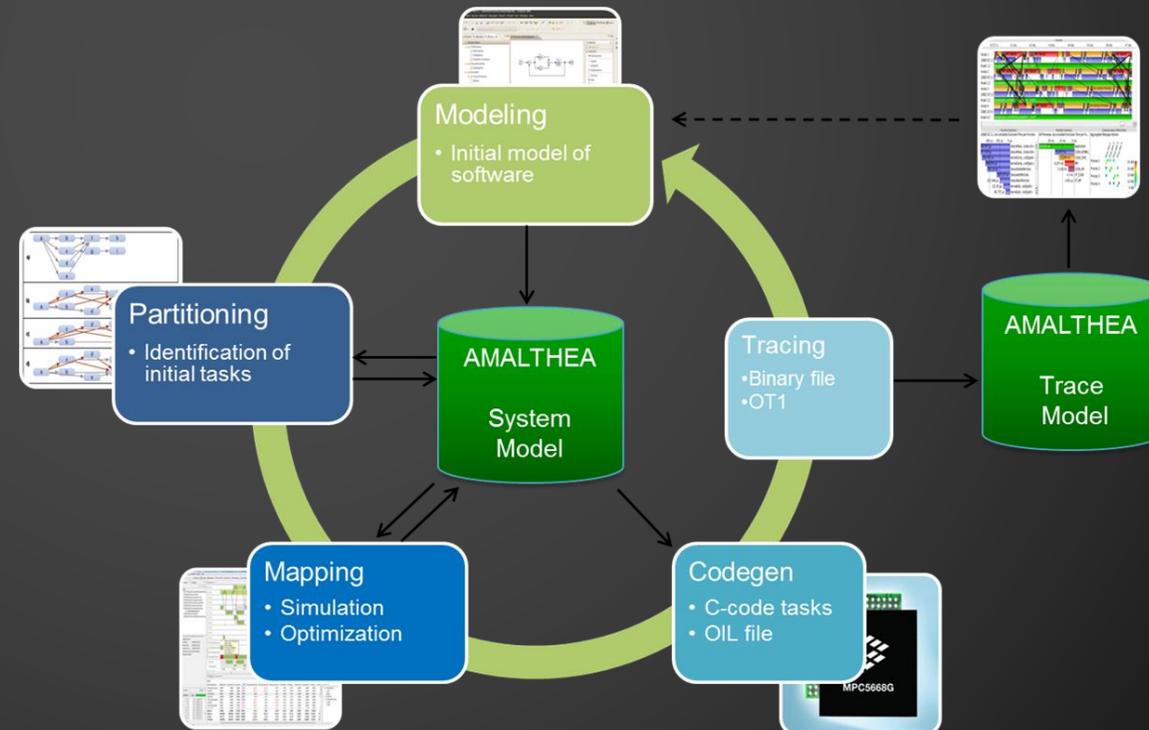


ARCHITECTURE MODELLING

- Model hardware – level depends on prediction needs
 - Transistors
 - Memory (cache, DRAM, cache policy)
 - Processor (pipelining, temperature, number of cores, frequency)
- Static code analysis
- Dynamic monitoring
- Perform analysis on models

AMALTHEA

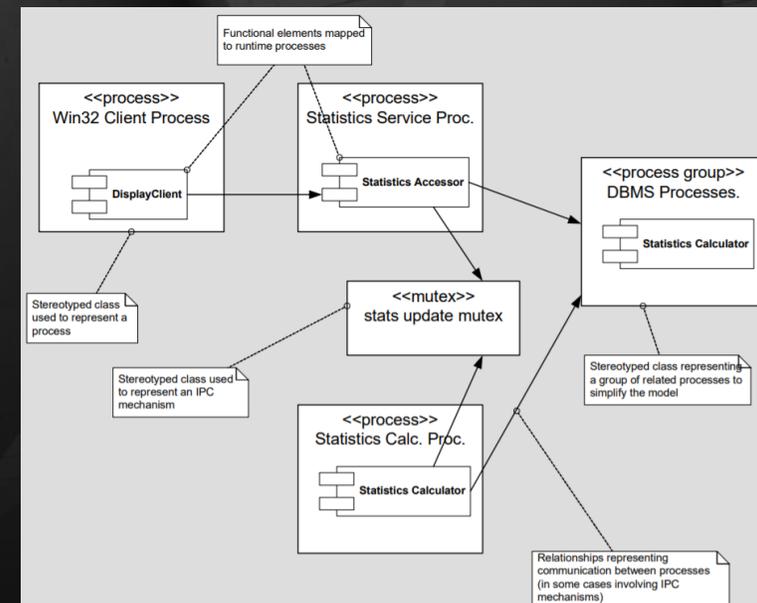
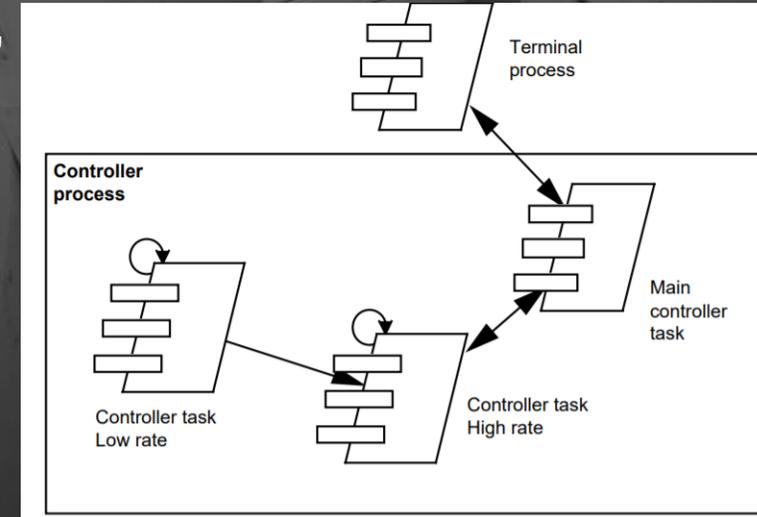
- Open source tool platform for engineering embedded multi- and many-core software systems
- <http://www.amalthea-project.org/>



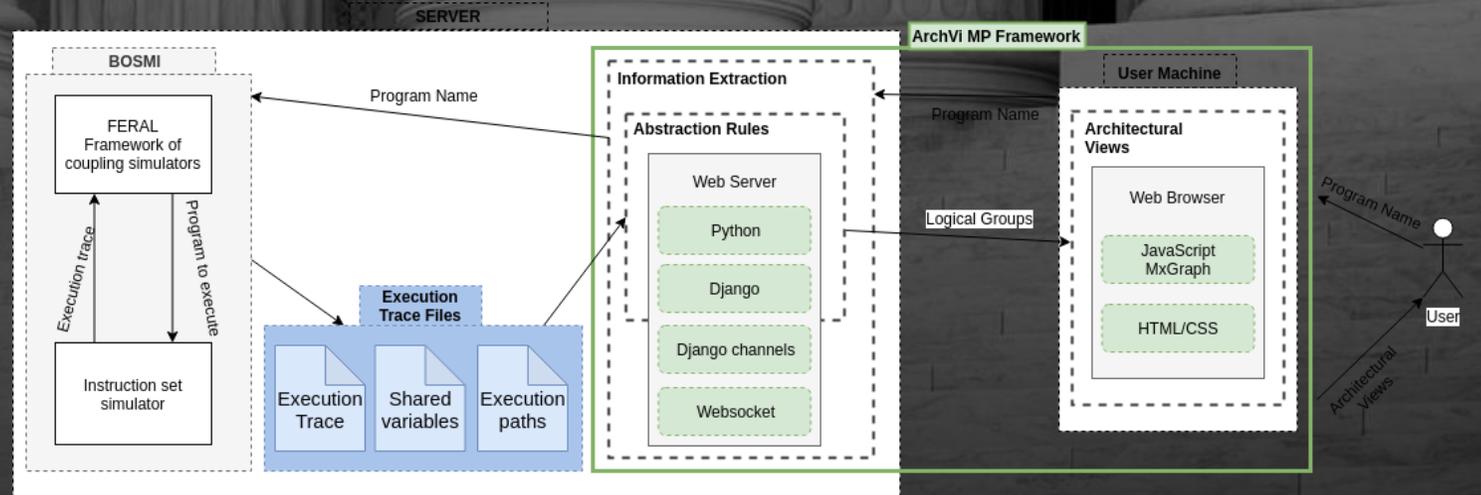
ARCHITECTURAL VIEWS FOR CONCURRENCY AND PARALLELISM

<https://www.viewpoints-and-perspectives.info/vpandp/wp-content/themes/secondedition/doc/spa191-viewpoints-and-perspectives.pdf>

- Process View - "4+1" view, P. B. Kruchten, "The 4+ 1 view model of architecture," IEEE software, vol. 12, no. 6, pp. 42–50, 1995
- Concurrency View, N. Rozanski and E. Woods, Software systems architecture: working with stakeholders using viewpoints and perspectives, 2nd ed. Upper Saddle River, NJ: Addison-Wesley, 2012.



ARCHITECTURAL
VIEWS FOR
MULTITHREADED
PROGRAMS - A
FRAMEWORK FOR
AUTOMATIC
EXTRACTION OF
CONCURRENCY-
RELATED
ARCHITECTURAL
PROPERTIES
FROM SOFTWARE



<https://mpourjafarian.github.io/ArchViMP.github.io/>

SIMULATORS

- SystemC
- Memory (e.g., DRAMSys: Tool for Optimizing Memory Systems through Simulation Analyses - https://www.iese.fraunhofer.de/en/innovation_trends/autonomous-systems/memtonomy/DRAMSys.html)
- The Sniper Multi-Core Simulator - [https://snipersim.org//w/The Sniper Multi-Core Simulator](https://snipersim.org//w/The_Sniper_Multi-Core_Simulator)
- gem5 - <https://www.gem5.org/>

Speedups from performance engineering a program that multiplies two 4096-by-4096 matrices. “Absolute speedup” is time relative to Python, and “relative speedup,” which we show with an additional digit of precision, is time relative to the preceding line.

IS CONCURRENT PROCESSING ON MULTICORES THE ANSWER TO OUR TROUBLES?

(4) parallelizing the code to run on all 18 of the processing cores, (5) exploiting the processor’s memory hierarchy, (6) vectorizing the code, and (7) using Intel’s special Advanced Vector Extensions (AVX) instructions.

	Implementation	Running time (s)	Absolute speedup	Relative speedup
1	Python	25 552.48 (~7 hours)	1	-
2	Java	2 372.68	11	10.8
3	C	542.67	47	4.4
4	Parallel loops	69.80	366	7.8
5	Parallel divide and conquer	3.80	6727	18.4
6	plus vectorization	1.10	23 224	3.5
7	plus AVX intrinsics	0.41	62 806	2.7

There’s plenty of room at the Top: What will drive computer performance after Moore’s law? E. Leiserson et al, Science 05 Jun 2020:Vol. 368, Issue 6495, DOI: 10.1126/science.aam9744

MANUAL VS AUTOMATIC PARALLELISATION

- **"Virtually every C++ application developed at Google is multithreaded."**, ThreadSanitizer – data race detection in practice, K. Serebryany, T. Iskhodzhanov, Workshop on Binary Instrumentation and Applications, 2009
- OpenMP
- An Implementation of LLVM Pass for Loop Parallelization Based on IR-Level Directives, K. Jingu et al., 2018
- Hydra - <https://github.com/jamrol149/Hydra>
- Janus - <https://github.com/timothymjones/Janus>
- SLX C/C++ - <https://www.silexica.com/products/slx-c/>

HETEROGENEOUS ARCHITECTURES



- Moore's law is still alive
- More transistors on the same surface
- More cores
- Increase in power consumption and heat dissipation (without frequency increases)
- Not all cores can be powered at the same time
- Dark silicon

HETEROGENEOUS ARCHITECTURES

*Unity in Diversity: Co-operative
Embedded Heterogeneous
Computing, Keynote, Tulika Mitra,
SAMOS 2018*

- Turning a problem into an opportunity
- Silicon area is cheaper relative to power
- Spend area to buy power
- Right core for the right task: Performance and Efficiency
- Missing piece: Software for heterogeneous
- Do we need to break HW-SW abstraction?



CONCLUSIONS

- Few drivers (set#1)
- Complex follow-up requirements (set#2,3,4)
- What is important and what is not
 - Scale and use case matter
- It is hard to make proper architectural decisions
- And...once you get the design right (Design Space Exploration – part 2) – you still need to develop and test it properly (part 3).

AGENDA



9:30

Session 1: Fundamental Issues with Concurrency in Embedded Software Systems from Architectural Point of View

10:30

10:45

Session 2: Modelling and DSE Methods for Mixed-Critical Software Systems using Multicore Architectures

11:45

12:00

Session 3: Synchronization in Concurrent Software is an Architectural Decision

13:00