Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Real-Time Systems

2-Basics

Antonio Camacho Santiago
antonio.camacho.santiago@upc.edu

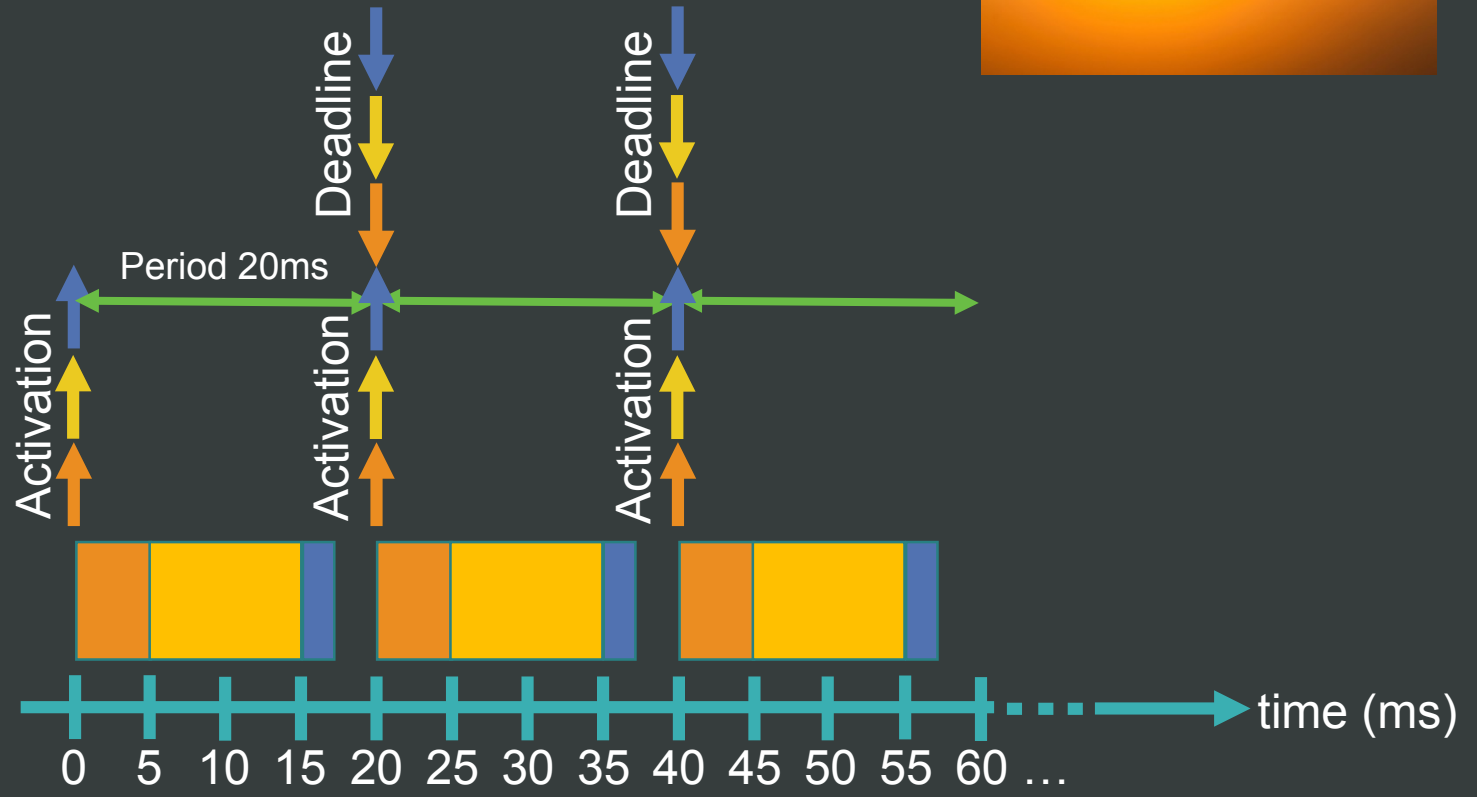
Drone flight

Application: Autonomous drone flight system

Requirements: flight stabilization by using periodic tasks

- Task 1: Read data form inertial measurement unit
- Task 2: Compute control
- Task 3: Update motor drivers
- Precedence $\tau_1 \rightarrow \tau_2 \rightarrow \tau_3$

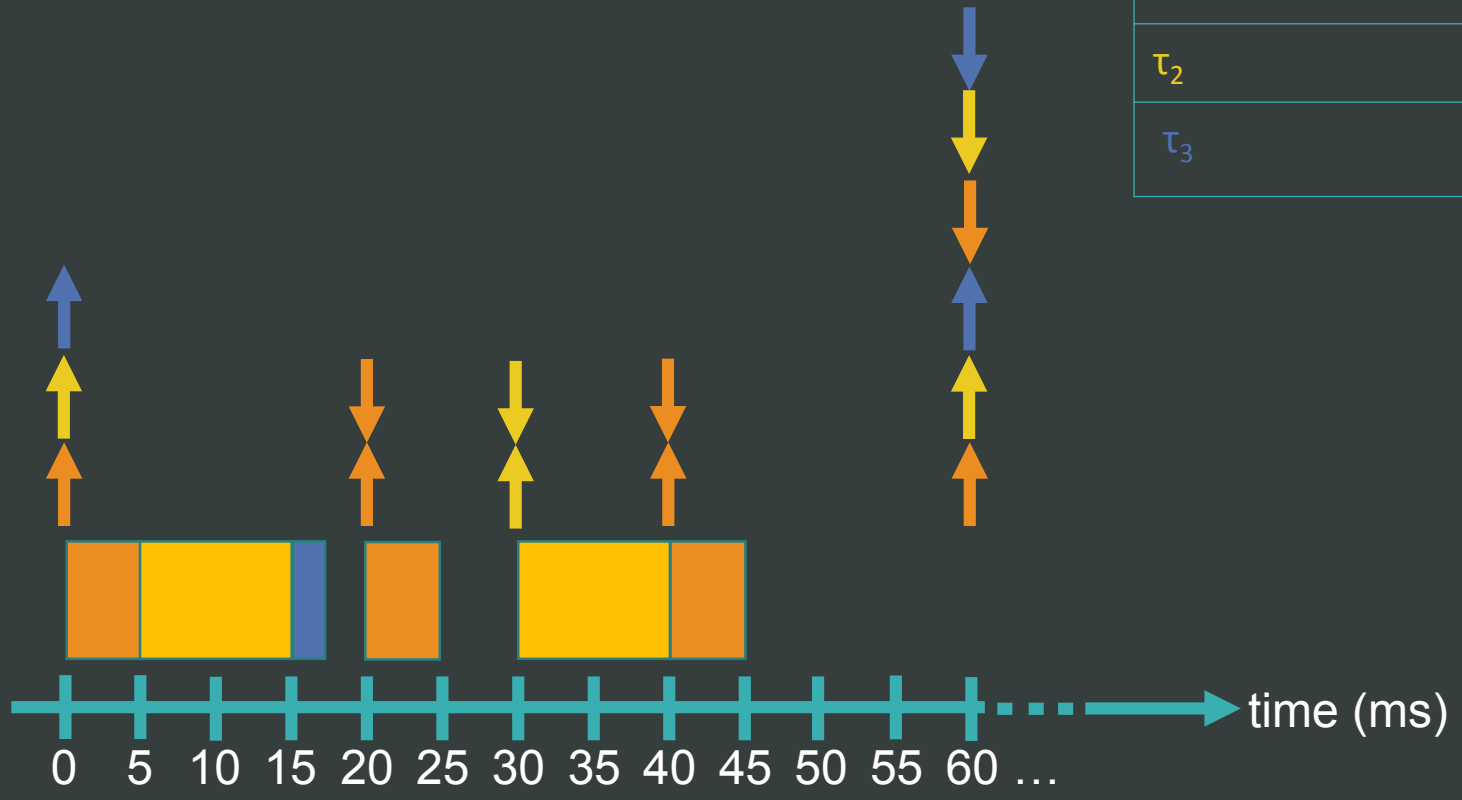
Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)
τ_1 sampling	5	20
τ_2 control	10	20
τ_3 update	2	20



Example

Can these tasks be scheduled?

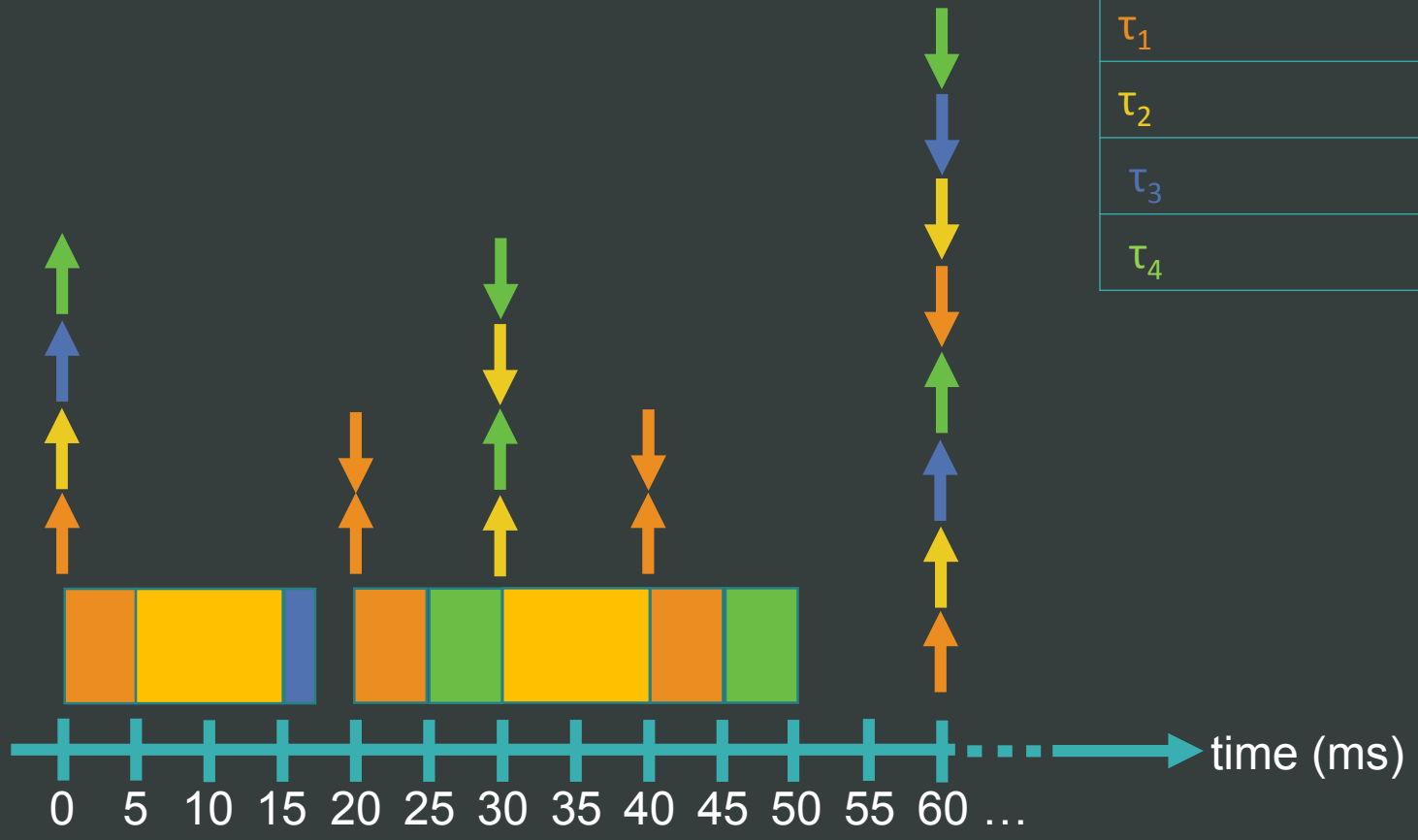
Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)
τ_1	5	20
τ_2	10	30
τ_3	2	60



Example

Can these tasks be scheduled?

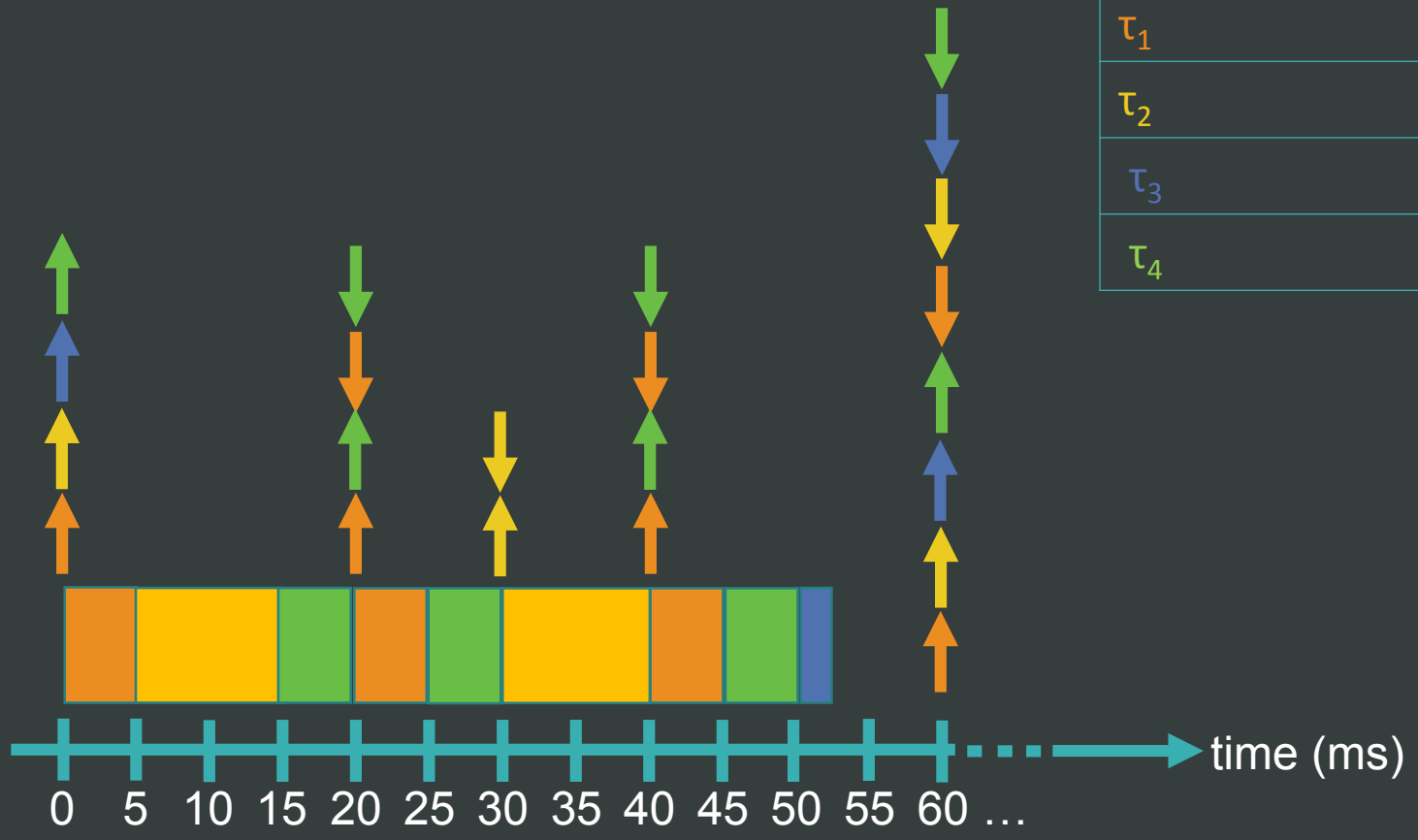
Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)
τ_1	5	20
τ_2	10	30
τ_3	2	60
τ_4	5	30



Example

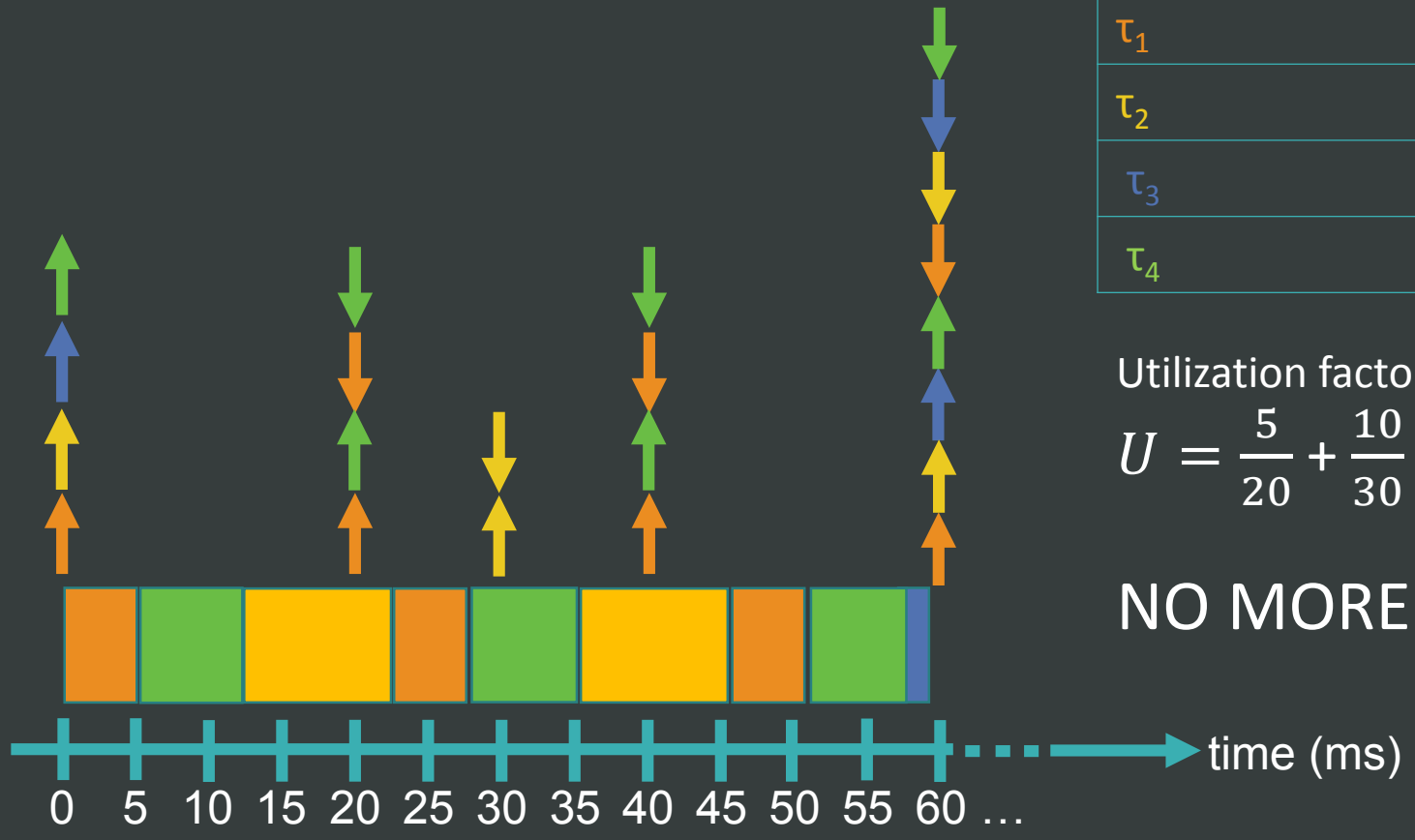
Can these tasks be scheduled?

Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)
τ_1	5	20
τ_2	10	30
τ_3	2	60
τ_4	5	**20**



Example

Can these tasks be scheduled?



Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)
τ_1	5	20
τ_2	10	30
τ_3	2	60
τ_4	**7.5**	20

Utilization factor:

$$U = \frac{5}{20} + \frac{10}{30} + \frac{2}{60} + \frac{7.5}{20} = 0.9917$$

NO MORE TASKS, PLEASE!!!

Utilization factor *U*

Processor overloaded?

$$U_i = \frac{c_i}{T_i} \text{ (for each task)}$$

$$U_{\text{total}} = \sum_{i=1}^{n} U_i = \sum_{i=1}^{n} \frac{c_i}{T_i} = \frac{c_1}{T_1} + \frac{c_2}{T_2} + \cdots + \frac{c_n}{T_n}$$

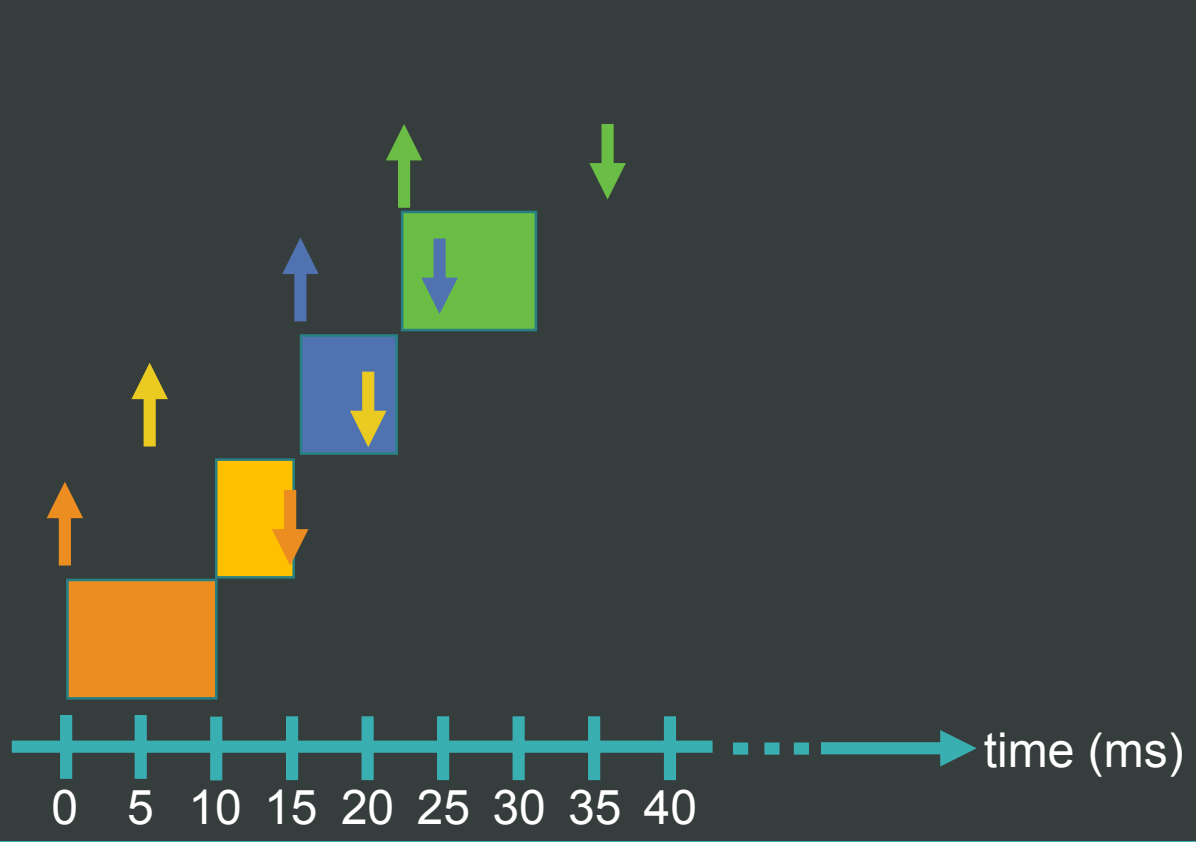
Task τ_i	c_i (ms)	T_i (ms)	Utilization factor U_i per unit (p.u.)
τ_1	5	20	0.250
τ_2	10	30	0.333
τ_3	2	60	0.033
τ_4	7.5	20	0.375
			0.9917

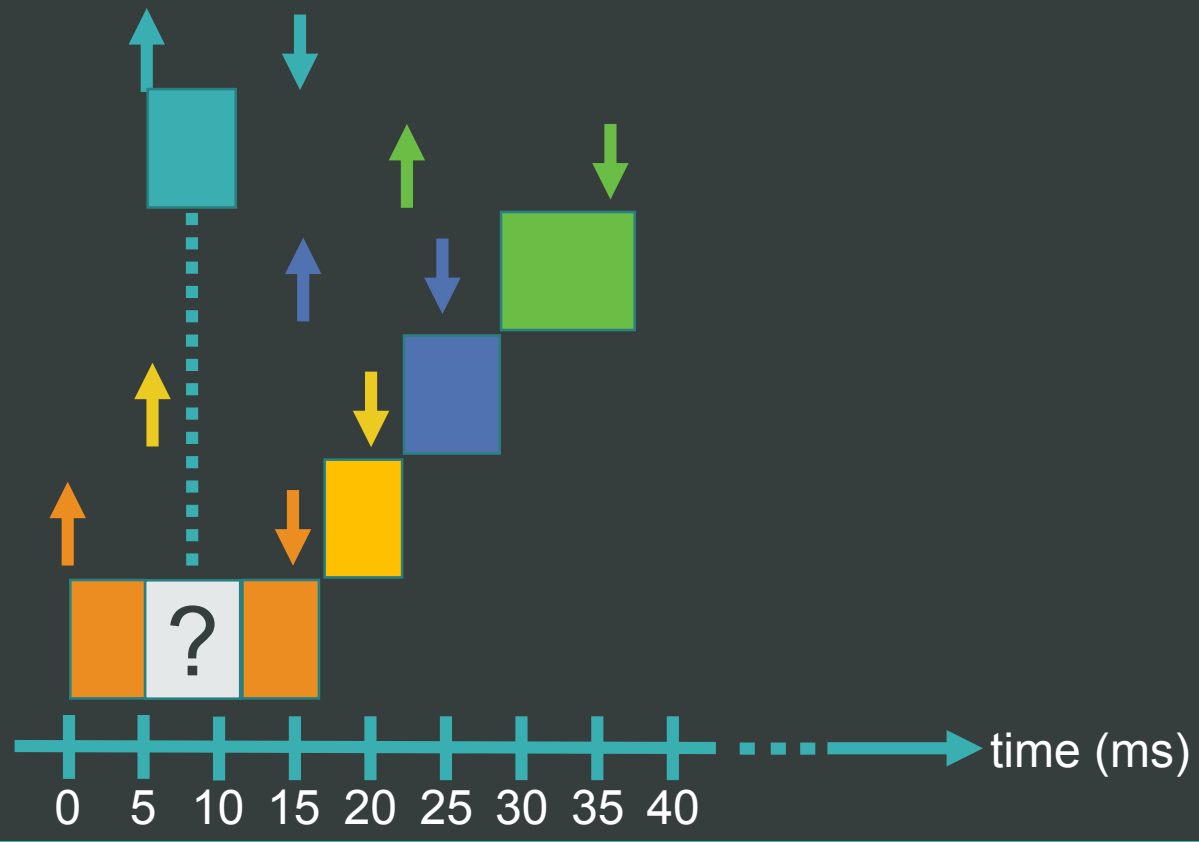
Necessary but not sufficient condition $U_{\text{total}} \leq 1$

Example

Can these tasks be scheduled?

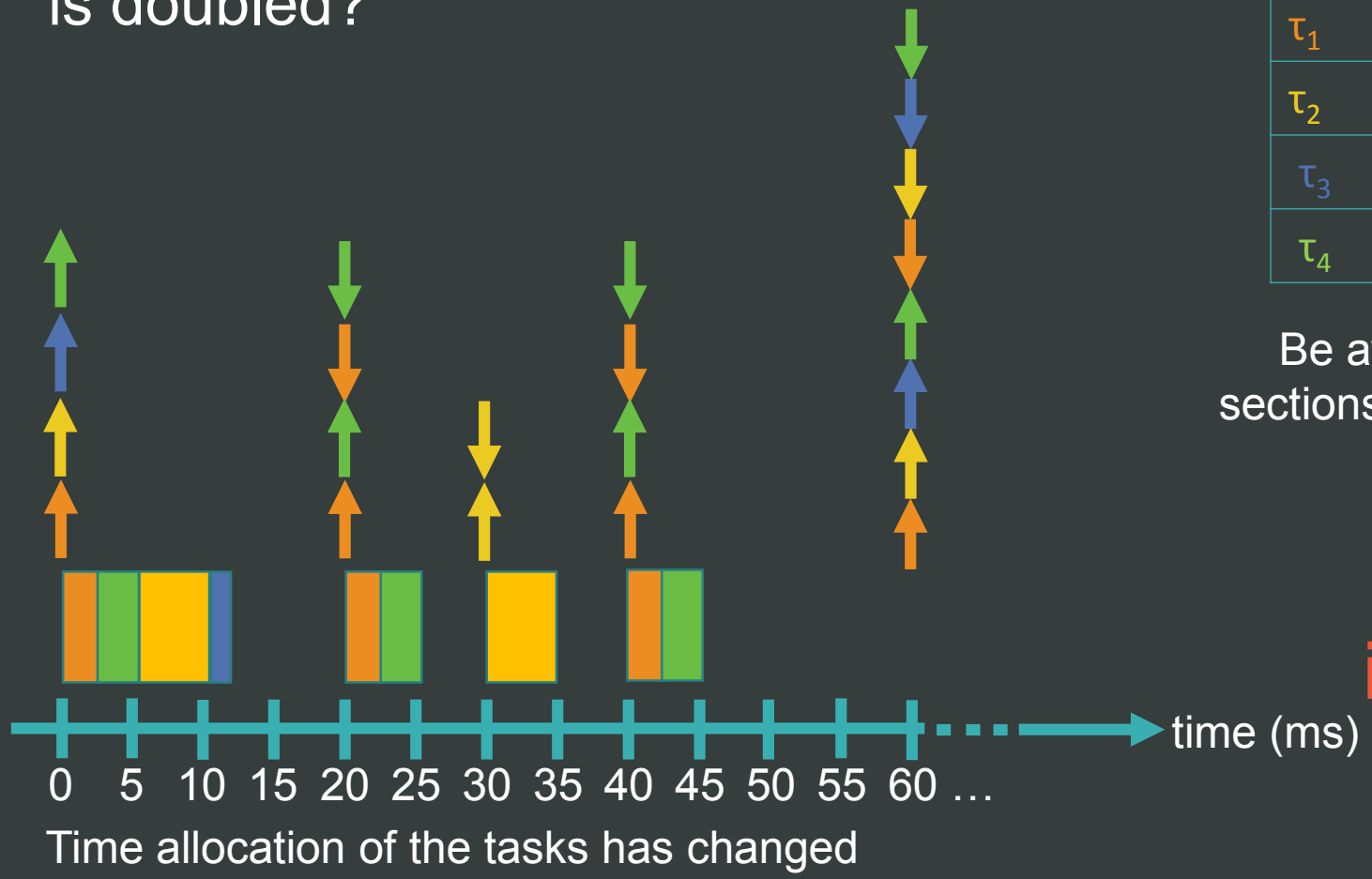
Domino effect… Something happens in task1 (a new task, preemption, blocking, exclusion, sporadic event, overload, ..) and all the tasks miss their deadlines





Faster processor

What happens if the processing speed is doubled?



Time allocation of the tasks has changed

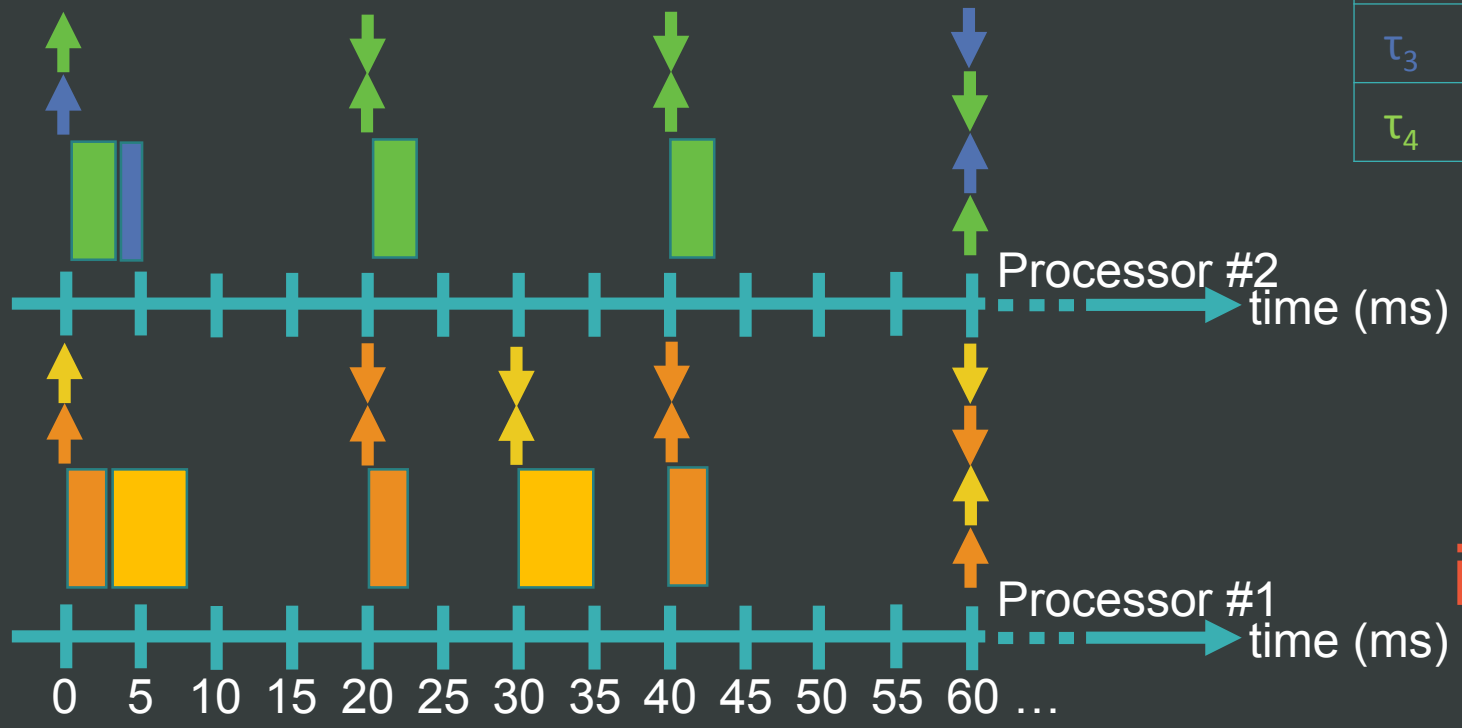
Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)
τ_1	5/2	20
τ_2	10/2	30
τ_3	2/2	60
τ_4	7.5/2	20

Be aware of context switches, entering critical sections, mutual exclusions, precedence, shared resources, SW and HW limitations, etc…

Probably it will work but it is mandatory to check everything again!

A new processor

What happens if a new processor is available?



Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)
τ_1	5/2	20
τ_2	10/2	30
τ_3	2/2	60
τ_4	7.5/2	20

Be aware of allocation, precedence, performance, inter-processor communications, etc…

Probably it will work but it is mandatory to check everything again!

Risk assessment

Probably it will work but it is
mandatory to check everything again!

Risk assessment: it is important to evaluate the risk associated with any change in the HW and SW components.

New analysis, understanding and tests need to be conducted under probably risky modifications.

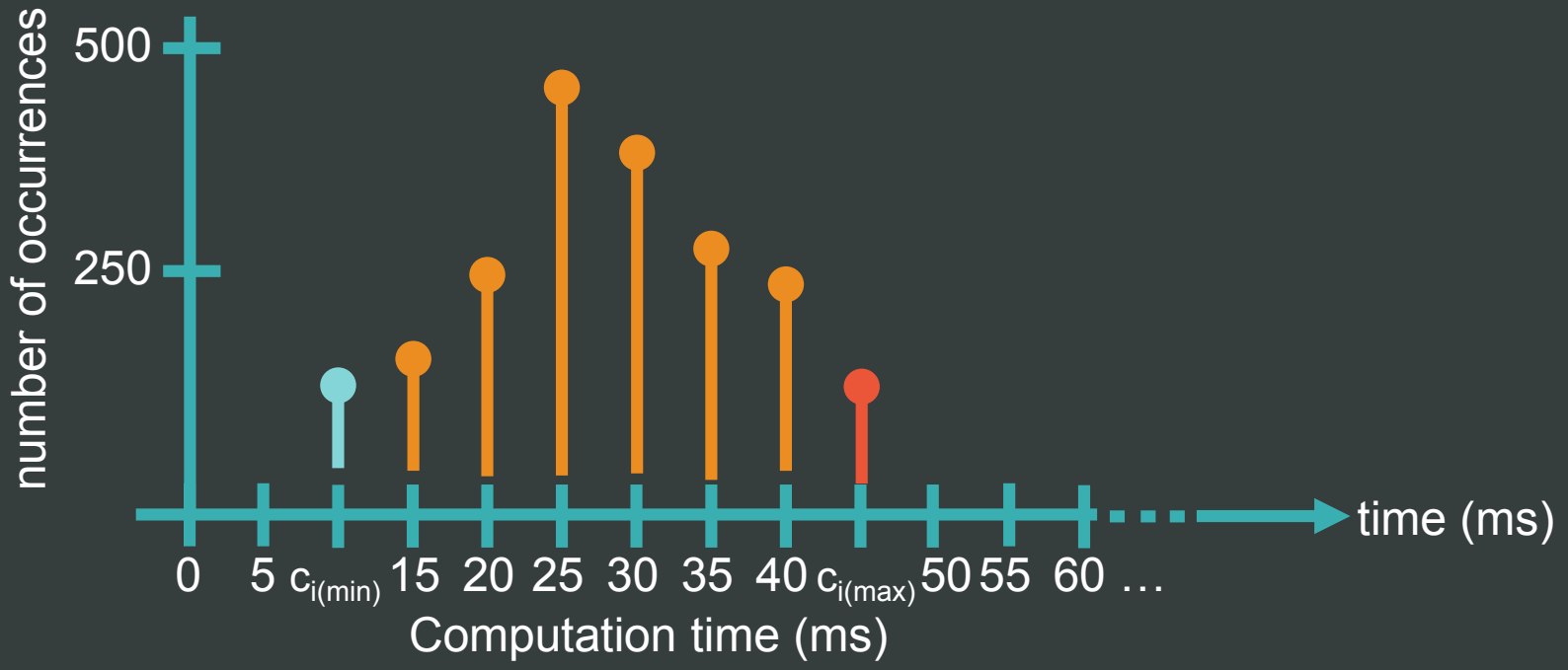
Pros.: Reduce issues, save costs, avoid malfunctioning, test new overload conditions, …

Cons.: Coverage tests, development time, human overload, …

Worst-Case Execution Time

How long does a SW task code take to complete?

```
while (y<3)
   if x>5 then
      x:=x+1;
   else
      x:=x*3;
   endif
```



Worst-Case Execution Time

How long does a SW task code take to complete?

You must know the WORST-CASE EXECUTION TIME to guarantee the deadlines of each task.

Remarks: *"The worst case execution time will always occur at the worst time"*

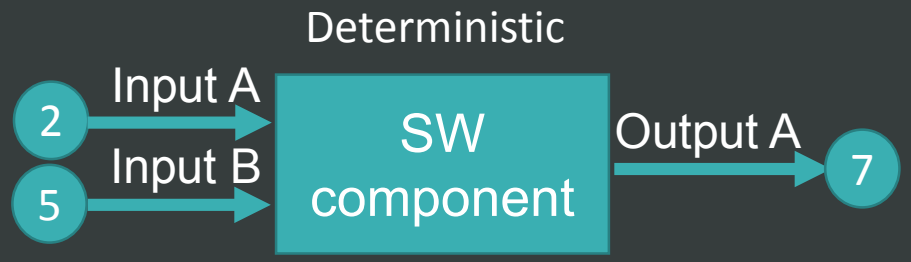
- Each job operates on different data `x` and `y` and can take different paths.
- Even for the same data, computation time depends on the processor (cache, prefetching, preemptions).
- Tests, debugging and measurements! It doesn't mean that it is guaranteed to have seen the longest time (e.g. Pathfinder issue!!!)
- Check hardware datasheets, handbooks, etc. and look for clock frequency, clock peripheral, phase lock loop, cycles/instruction, etc. You must know how time is managed
- Check programming/compilers datasheets. In the disassembly list, identify mnemonics and split them into small code blocks. This is not 100% safe. This is not straight-forward.
- Code coverage and mocking tools to find the worst path of execution

Worst-Case Execution Time: Determinism

How long does a SW task code take to complete?

Determinism:

In computer science, a deterministic algorithm is an algorithm which, given a particular input, will always produce the same output, with the underlying machine always passing through the same sequence of states.



Worst-Case Execution Time: Nmemonics

How long does a SW task code take to complete?

```
MOV X,D0 //move source->dest
CMP D0,5 //compare
BGT L1   //branch if greater than 0
MUL D0,3 //multiply
JMP L2   //jump
L1: ADD D0,1 //label 1 add
L2: ...  //label 2
```

Mnemonic	Number of cpu cycles
MOVE	8
CMP	4
BGT	4
MUL	16 or 32**
JMP	4
ADD	4

* It depends on some rules that no one can understand

WCET= {analysis of the cases} = n cycles @ 8MHz = # µs

Worst-Case Execution Time: Debugging

How long does a SW task code take to complete?

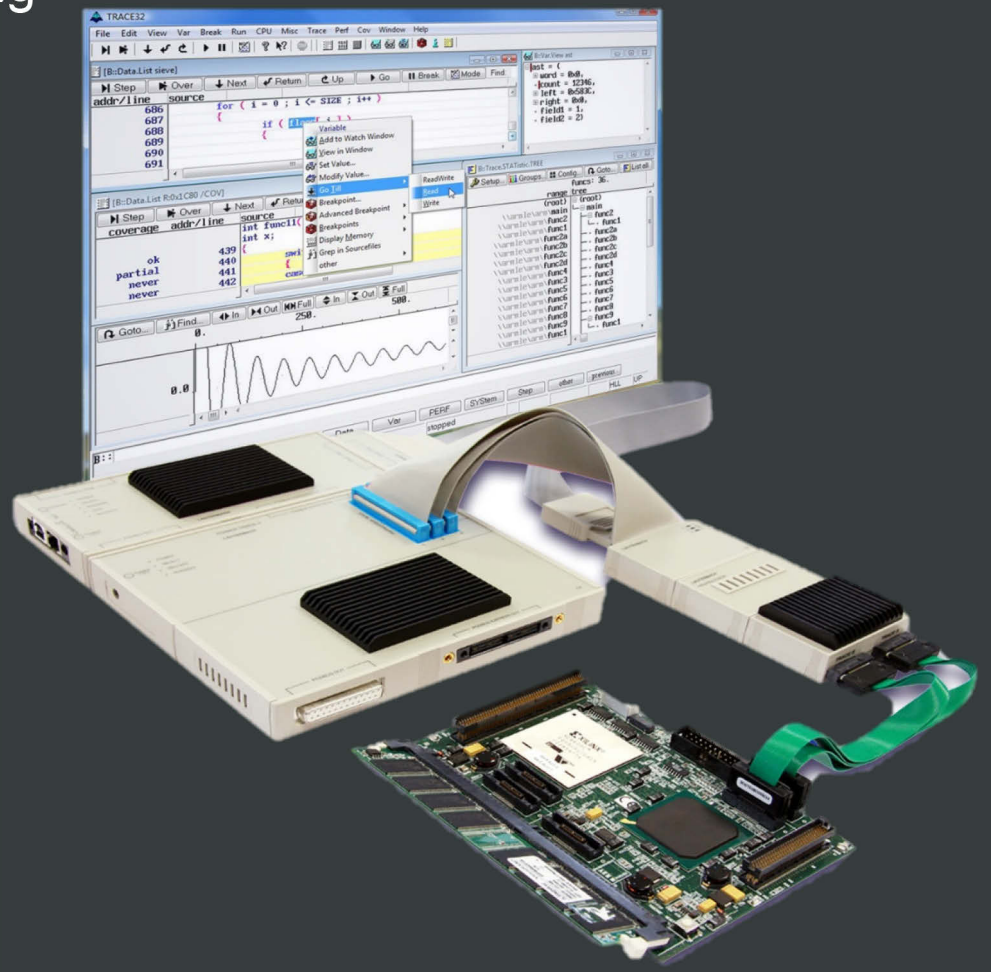
How to analyze the path for each SW task → Debugging

FTDI-based chips:

- Low cost, low performance
- Virtual COM+USB2JTAG

Trace 32 + Lauterbach:

- Multi-platform
- API for custom automated tests
- High cost, high performance



Worst-Case Execution Time: Architecture

How long does a SW task code take to complete?

It depends on the microcontroller architecture:

1 instruction = x clock cycles

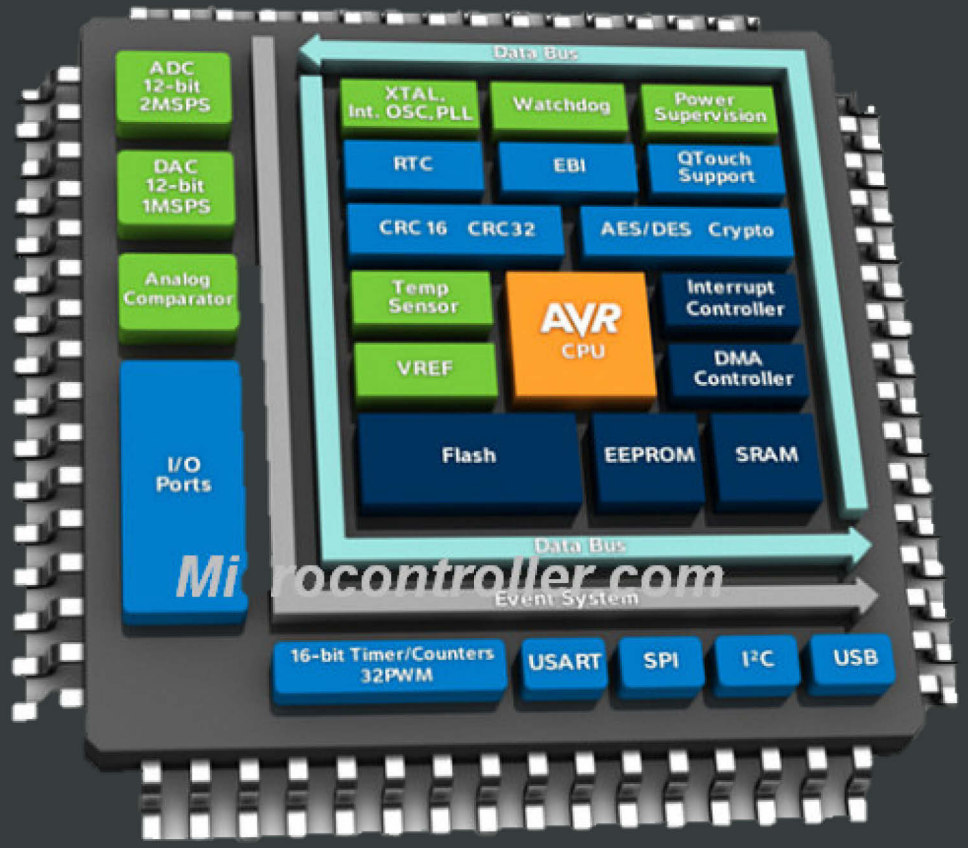
MIPS (Mega instructions per seconds) vs clock frequency

ALU: Fixed/floating point calculations

Math support for trigonometric functions

Peripherals:

- encoder
- sin-cos peripheral
- control law accelerator
- PWM
- communications
- complex drivers



Worst-Case Execution Time: Preemption

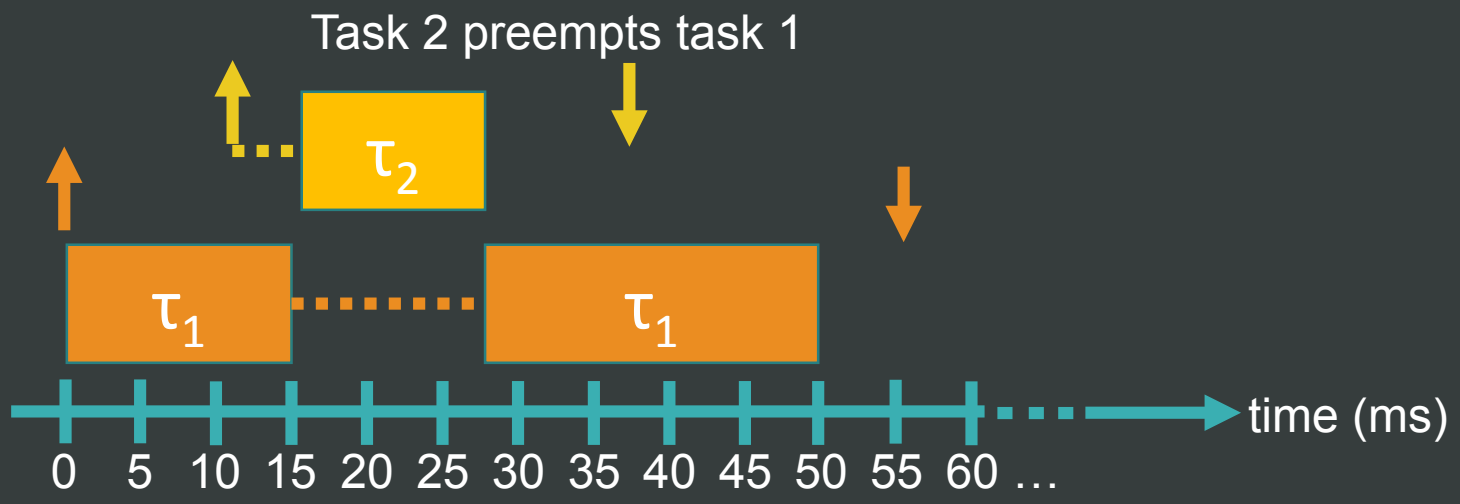
How long does a SW task code take to complete?

PREEMTION: It is a kernel mechanism that allows to suspend the execution of the running task in favor of a more important task.

Preemption helps concurrency and allows reducing the response times of high priority tasks.

Preemption destroys program locality and introduces runtime overhead

It can be disabled (completely or temporarily) to ensure the consistency of certain critical operations.



Worst-Case Execution Time: Exclusion

How long does a SW task code take to complete?

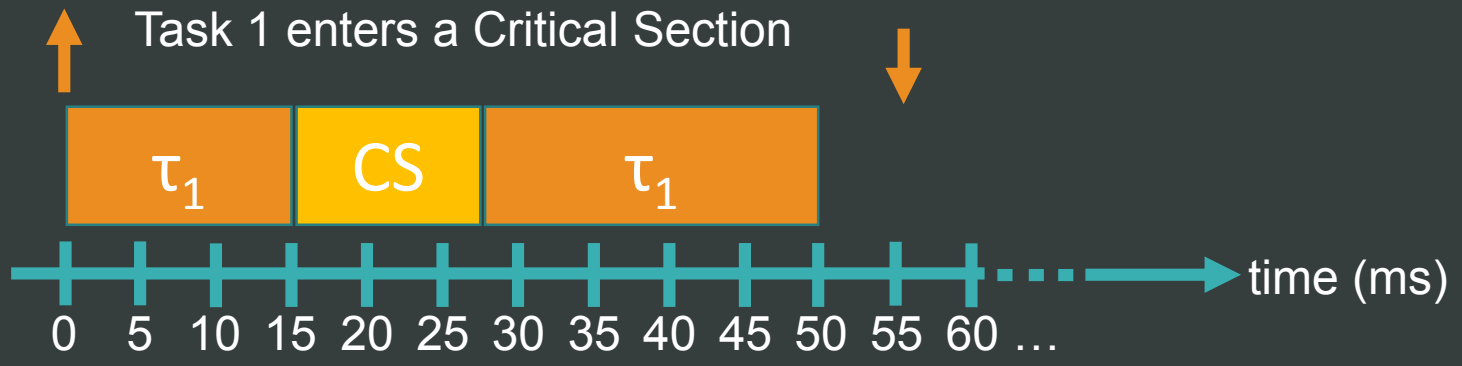
MUTUAL EXCLUSION need to be accounted for

Some resources are inherently non-shareable and require special attention in RT-systems

To preserve data consistency, shared resources must be accessed in mutual exclusion.

However, mutual exclusion introduces extra delays

Entering a Critical Section can be a solution. One Publisher, multiple Subscribers in a synchronous fashion can also did the job in a different manner. The best solution depends on the shared resource

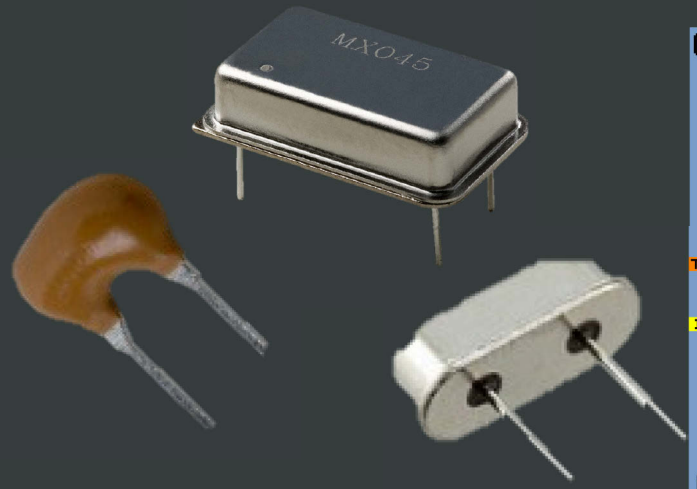


Worst-Case Execution Time: Clock drift

How long does a SW task code take to complete?

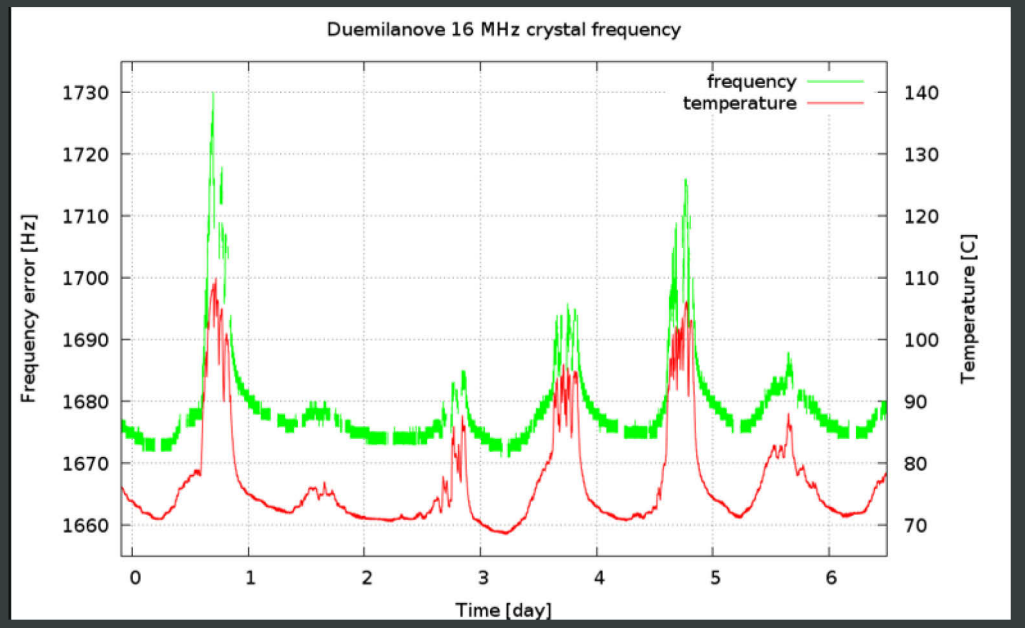
Moreover, execution time depends on HW clocks!

Clocks drift due to temperature, age and acceleration



Resonator, oscillator and crystal for clock signal generation

Signal attached to the clock input pins of a microcontroller



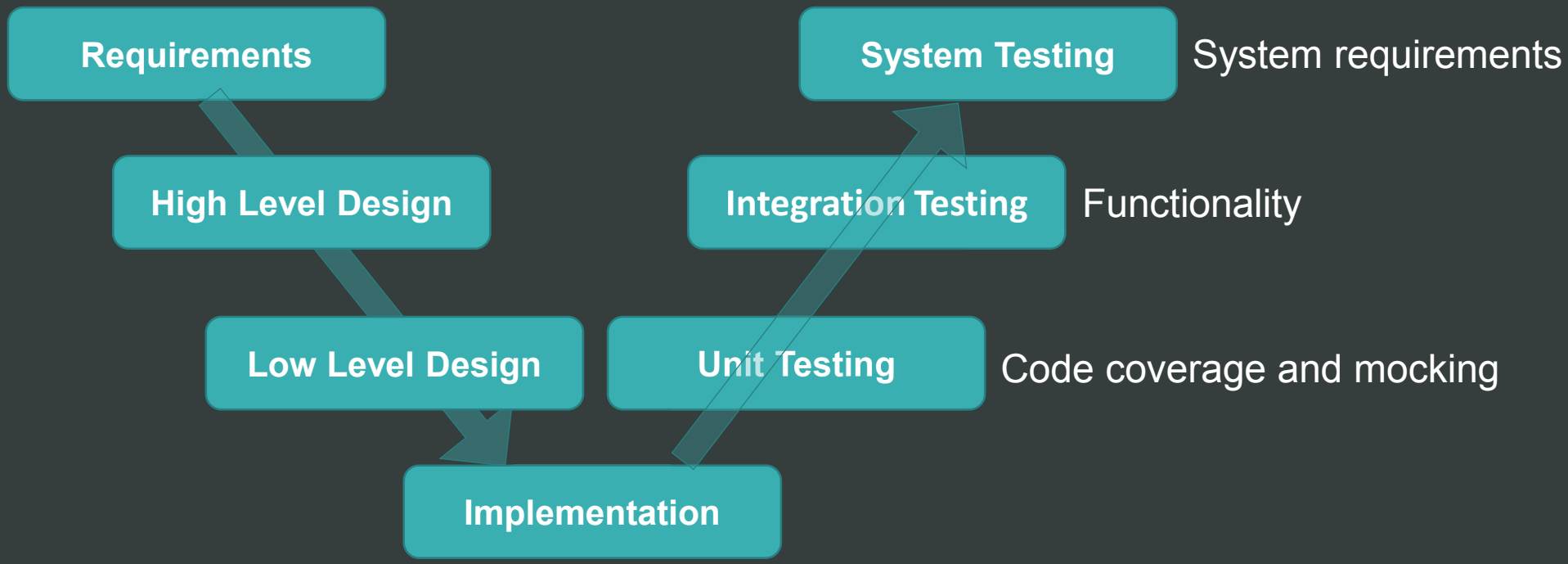
Clock frequency as a function of temperature

Worst-Case Execution Time: SW analysis

How long does a SW task code take to complete?

How to analyze the path for each SW task → Code coverage and mocking tools

V-model for SW development in automotive



Worst-Case Execution Time: Code coverage

How long does a SW task code take to complete?

How to analyze the path for each SW task → Code coverage tools

Code coverage: check all paths in a SW code to check all conditions and all lines of code

Sw code to be tested:

```
function roots = quadraticSolver(a,b,c)
% quadraticSolver returns solutions to the
% quadratic equation a*x^2 + b*x + c = 0.

checkInputs

roots(1) = (-b + sqrt(b^2 - 4*a*c)) / (2*a);
roots(2) = (-b - sqrt(b^2 - 4*a*c)) / (2*a);

    function checkInputs
        if ~isa(a,'numeric') || ~isa(b,'numeric') || ~isa(c,'numeric')
            error('quadraticSolver:InputMustBeNumeric', 'Coefficients must be numeric.')
        end
    end
end
```

Worst-Case Execution Time: Code coverage

How long does a SW task code take to complete?

How to analyze the path for each SW task → Code coverage tools

Code coverage: check all paths in a SW code to check all conditions and all lines of code

Sw tests:

```
classdef SolverTest < matlab.unittest.TestCase
    % SolverTest tests solutions to the quadratic equation
    % a*x^2 + b*x + c = 0

    methods (Test)
        function testRealSolution(testCase)
            actSolution = quadraticSolver(1,-3,2);
            expSolution = [2,1];
            testCase.verifyEqual(actSolution,expSolution)
        end
        function testImaginarySolution(testCase)
            actSolution = quadraticSolver(1,2,10);
            expSolution = [-1+3i, -1-3i];
            testCase.verifyEqual(actSolution,expSolution)
        end
        function testInfSolution(testCase)
            actSolution = quadraticSolver(Inf,Inf,Inf);
            expSolution = [NaN, NaN];
            testCase.verifyEqual(actSolution,expSolution)
        end
    end
end
```

Worst-Case Execution Time: Code coverage

How long does a SW task code take to complete?

How to analyze the path for each SW task → Code coverage tools

Code coverage: check all paths in a SW code to check all conditions and all lines of code

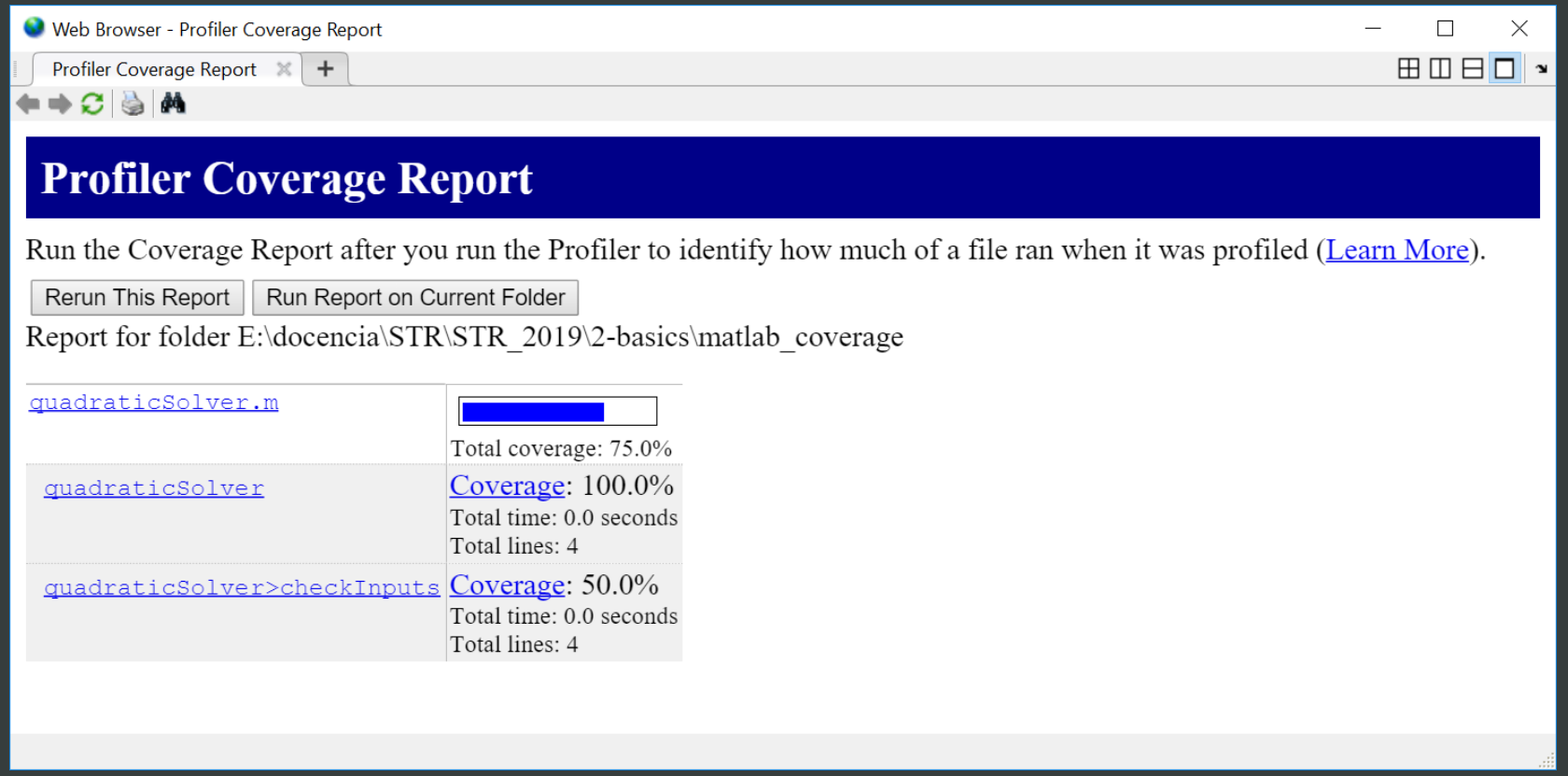
Results:

Code coverage tools:

Java: EMMA, Cobertura

C: C Test

Python: Coverage.py



Worst-Case Execution Time: Mocking

How long does a SW task code take to complete?

How to analyze the path for each SW task → Mocking tools

In object-oriented programming, mock objects are simulated objects that mimic the behavior of real objects in controlled ways. A programmer typically creates a mock object to test the behavior of some other object

Mocking tools:

Java: Mockito, JMock, EasyMock

JavaScript: MockusJS

C: Moq, Rhino Mocks

Python: Mock for Python 3.0