Design and Analysis of Time-Critical Systems
Introduction

Jan Reineke @ SAARLAND UNIVERSITY
COMPUTER SCIENCE

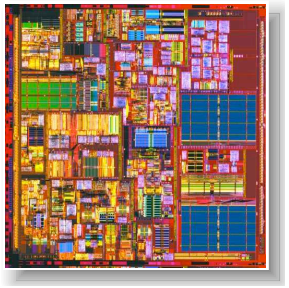
ACACES Summer School 2017
Fiuggi, Italy

Structure of this Course

2. How are they implemented? (briefly)


```
// Perform the convolution.
for (int i=0; i<10; i++) {
  x[i] = a[i]*b[j-i];
  // Notify listeners.
  notify(x[i]);
}
```

+



?

1. What are real-time systems?

4. How to design timing-predictable microarchitectures?

3. How to verify the real-time constraints?

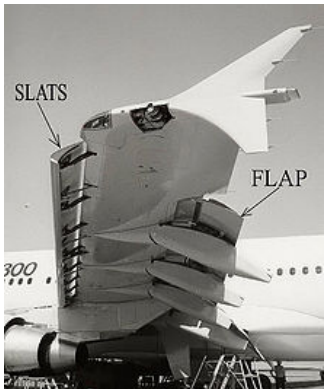
1. What are Real-time Systems?

In a *real-time system*, correctness not only depends on the logical results but also on the *time* at which results are produced.

- Typical misconception:
  - Real-time computing ≠ compute things *as fast as possible*
  - Real-time computing = compute *as fast as necessary*, *but also not too fast*

1. What are Real-Time Systems?

- Real-time systems are often embedded control systems
- Timing requirements often dictated by interaction with physical environment:
  - Examples in automotives:
    - ABS: Anti-lock braking systems
    - ESP: Electronic stability control
    - Airbag controllers
  - Many more examples in trains, avionics, and robotics…



YouTube

Classification of Real-time Systems Hard and Soft

"A real-time constraint is called **hard**, if not meeting that constraint could result in a catastrophe" [Kopetz, 1997]

- ➔ Safety-critical real-time systems
- ➔ Main focus of this course

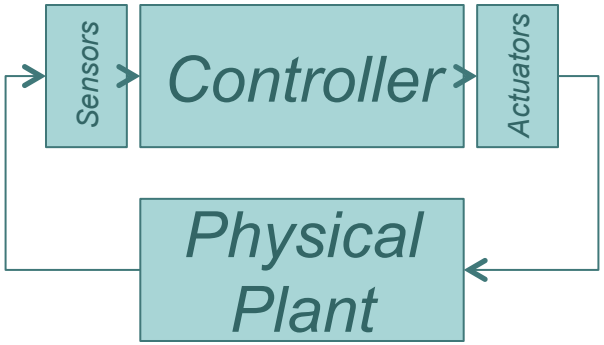
All other time constraints are called **soft**.

"A guaranteed system response has to be explained *without statistical arguments"* [Kopetz, 1997].

- ➔ Focus is on **safe, static** methods

2. How are they implemented?

Typical structure of control systems:



A very basic approach to program such a system:

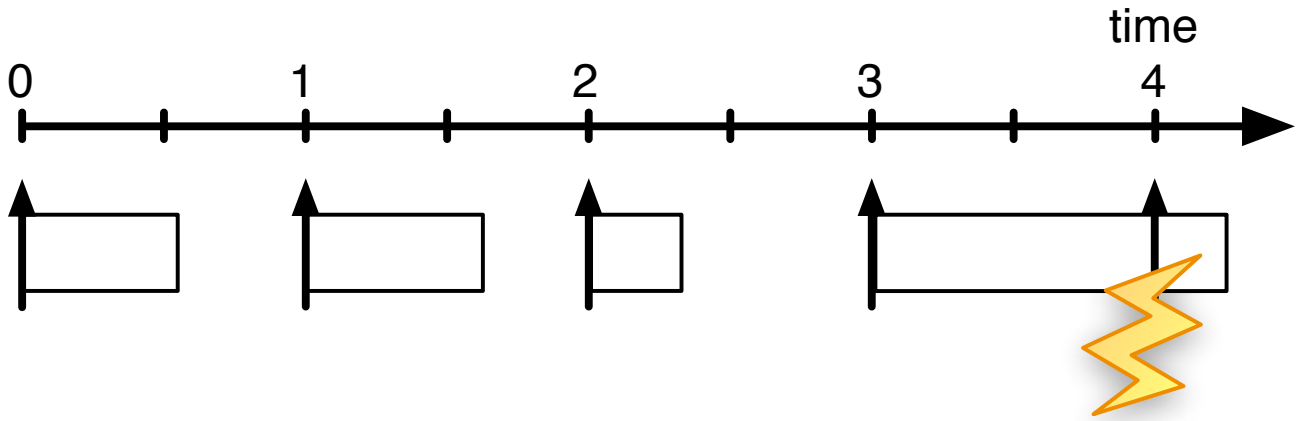
```
initialize state;
every clock tick do
      read inputs;
      compute outputs and next state;
      emit outputs
end-do
```

Can be modeled by automaton.

How to describe such automata?
→ Synchronous Languages

Basic Approach: Advantages

- Perfect match for sampled-data control theory
- Easy to implement, even on “bare” machine
- Timing analysis is comparably “simple”:



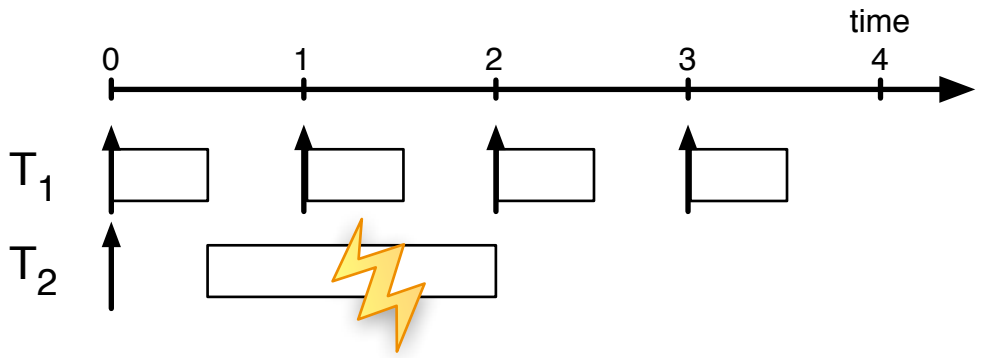
➔ *Need to make sure that:*
Worst-case execution time (WCET) < Period

2. How are they implemented? Preemptive Scheduling

What if different computations need to be performed at different periods?

Preemptive scheduling:

Non-preemptive execution of two periodic tasks:

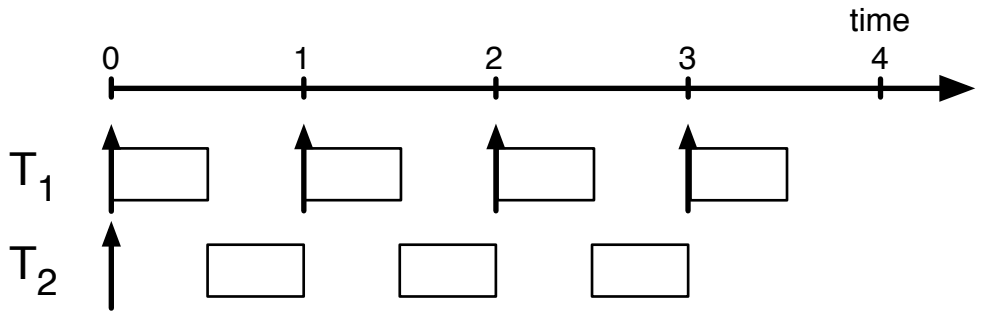


2. How are they implemented? Preemptive Scheduling

What if different computations need to be performed at different periods?

Preemptive scheduling:

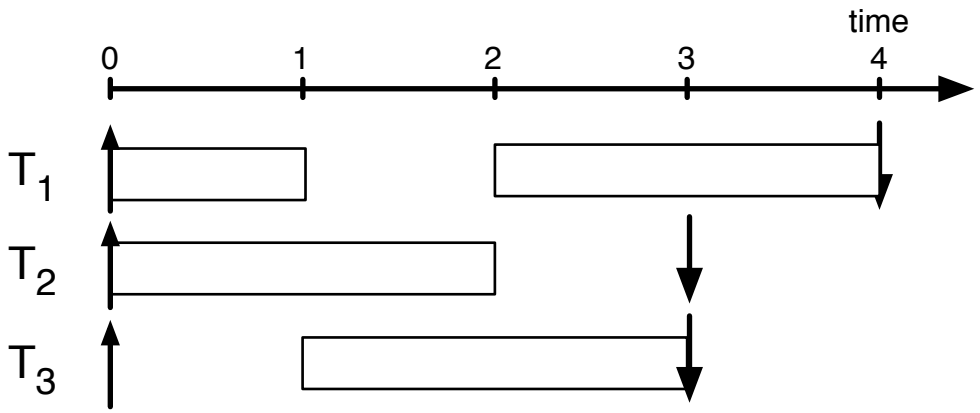
Preemptive execution of the two tasks:



2. How are they implemented? Multiprocessor Scheduling

What if we have a multi-core at our disposal?

Multiprocessor scheduling:



3. How to verify the real-time constraints? Timing Analysis

Traditional separation into two phases:

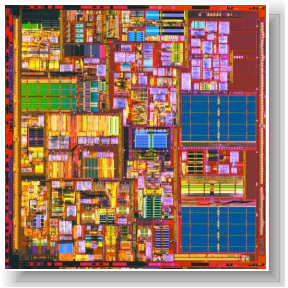
1. Worst-case Execution Time (WCET) Analysis
   determines bounds on the execution time of a task when run in isolation
2. Response-time Analysis
   determines bounds on tasks' response times given WCET bounds, accounting for interference due to scheduling decisions

3. How to verify the real-time constraints? WCET Analysis

Worst-case execution time = maximum execution time of a program on a given microarchitecture

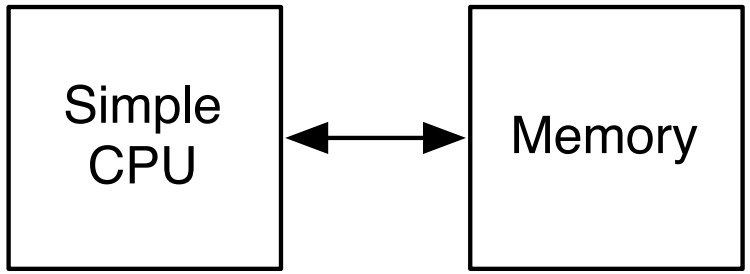
```
// Perform the convolution.
for (int i=0; i<10; i++) {
  x[i] = a[i]*b[j-i];
  // Notify listeners.
  notify(x[i]);
}
```

+



What does the execution time depend on?

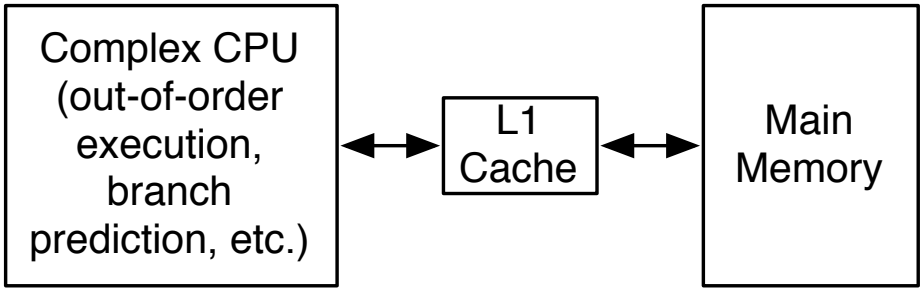
- The input, determining which path is taken through the program.



***Challenge**:*
How to determine feasible paths through a given program?

What does the execution time depend on?

- The input, determining which path is taken through the program.
- The state of the hardware platform:
  - Due to caches, pipelining, speculation, etc.



***Challenge**:*
How to determine feasible „microarchitectural paths“ through a given program?

Example of Influence of Microarchitectural State: Caches

x=a+b; →

```
LOAD   r2, _a
LOAD   r1, _b
ADD    r3,r2,r1
```

PowerPC 755
"flies" in Airbus planes

***Challenge 1a**:*
How to precisely and efficiently analyze cache behavior?

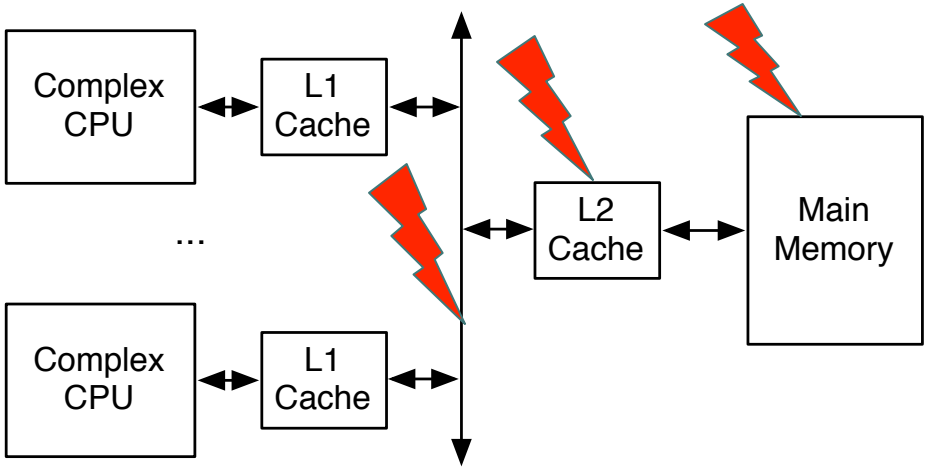
***Challenge 1b**:*
How to account for cache-related preemption delays?

***Challenge 2**:*
How to design caches and memory hierarchy to enable precise and efficient analysis?

Preemptions are not free!

What does the execution time depend on?

- The input, determining which path is taken through the program.
- The state of the hardware platform:
  - Due to caches, pipelining, speculation, etc.
- Interference from the environment:
  - External interference as seen from the analyzed task on shared busses, caches, memory.



Example of Influence of Corunning Tasks in Multicores

Radojkovic et al. (ACM TACO, 2012) on Intel Atom and Intel Core 2 Quad:

up to 14x slow-down due to interference on shared L2 cache and memory controller

***Challenge 1**:*
How to manage shared resources in a multicore?

***Challenge 2**:*
How to structure timing analysis (WCET + response-time analysis) in the presence of interference on shared resources?

Overview of Remainder of this Course

- WCET/Low-level Analysis
  - Basic Structure of Modern WCET Analyzers
  - Static Cache Analysis + Cache Predictability
- Response-time Analysis with a Focus on Shared Resources
  - Single Cores with Cache-related Preemption Overheads
  - Multi Cores with Shared Resources
- Design for Timing Predictability
  - Predictability and Analyzability
  - Case Studies:
    - Academic: Precision-Timed ARM (PTARM)
    - Industrial: Kalray MPPA-256