CHALMERS | UNIVERSITY OF GOTHENBURG

Real-Time Systems

7.5 credit points

Professor Jan Jonsson

Department of Computer Science and Engineering
Chalmers University of Technology

CHALMERS | UNIVERSITY OF GOTHENBURG

Course organization

Lectures (15 of them)

Lectures are offered in full class at least once a week, and with the goal of introducing the programming paradigm and basic scheduling theory as well as demonstrating how the paradigm and theory are applied in practice.

Special sessions (5 of them)

Special sessions are offered in full class on certain weeks, and should be seen as a complement to the lectures. Examples:

– help with software design and development tools
– introduction to sound generation and music theory
– discuss solutions to exercise problems or old exam problems

CHALMERS | UNIVERSITY OF GOTHENBURG

Course organization

Exercise sessions (7 of them)

Exercise sessions are offered in full class once a week (except on week 8), and with the goal of further exploring aspects of the laboratory assignment and the scheduling theory.

Laboratory assignment (one big assignment)

The compulsory assignment is done in smaller groups and will give the student practical experience with programming of a time-critical embedded system, using the C programming language and the TinyTimber kernel. The application to be considered is a distributed synthetic chorus with real-time tone generators.

The laboratory sessions run continuously from week 2 to week 7.

CHALMERS | UNIVERSITY OF GOTHENBURG

Course organization

Preliminary schedule

	Event	Time	Room	Subject	Links
Week 1					**Jan 15 - Jan 19**
Mon					
Tue	Lecture	13.15 - 15.00	*HC3*	Real-time systems: characteristics and design methods	notes
Wed					
Thu	Lecture	13.15 - 15.00	*HC3*	Real-time systems: programming paradigms	notes
Thu	Lecture	15.15 - 17.00	*HC3*	The TinyTimber kernel	notes
Fri	Exercise	13.15 - 15.00	*HC3*	Laboratory assignment: development system and target hardware	notes
Week 2					**Jan 22 - Jan 26**
Mon	*Laboratory session*				
Tue	*Laboratory session*				
Tue	Lecture	13.15 - 15.00	*HC3*	Concurrent programming: problems and solutions	notes
Wed	*Laboratory sessions*				
Thu	Lecture	13.15 - 15.00	*HC3*	Concurrent programming: problems and solutions (cont'd)	notes
Thu	Exercise	15.15 - 17.00	*HC3*	Programming with the TinyTimber kernel	notes
Fri	*Special*	13.15 - 15.00	*HC3*	Consultation session - laboratory assignment	
Week 3					**Jan 29 - Feb 2**
Mon	*Laboratory session*				
Tue	*Laboratory session*				
Tue	Lecture	13.15 - 15.00	*HC3*	Concurrent programming: guaranteeing timeliness	notes
Wed	*Laboratory sessions*				
Thu	Exercise	13.15 - 15.00	*HC3*	Programming with the TinyTimber kernel	notes
Thu	*Special*	15.15 - 17.00	*HC3*	Consultation session – laboratory assignment	
Fri	Lecture	13.15 - 15.00	*HC3*	Task model; Worst-case execution time	notes
Week 4					**Feb 5 - Feb 9**
Mon	*Laboratory session*				
Tue	*Laboratory sessions (extra evening session to replace Wed morning session)*				
Tue	*No lecture (due to CHARM)*				
Wed	*Laboratory session (evening only, due to CHARM)*				
Thu	Exercise	13.15 - 15.00	*HC3*	Worst-case execution time analysis	notes
Thu	*Special*	15.15 - 17.00	*HC3*	Consultation session – laboratory assignment	
Fri	Lecture	13.15 - 15.00	*HC3*	Real-time network communication	notes

CHALMERS | UNIVERSITY OF GOTHENBURG

Course organization

Week 5					**Feb 12 - Feb 16**
Mon	*Laboratory session*				
Tue	*Laboratory session*				
Tue	Lecture	13.15 - 15.00	*HC3*	Scheduling: general concepts and performance aspects	notes
Wed	*Laboratory sessions*				
Thu	*No lectures (due to Kårens dag)*				
Fri	Lecture	13.15 - 15.00	*HC3*	Scheduling: cyclic executives	notes
Week 6					**Feb 19 - Feb 23**
Mon	*Laboratory session*				
Tue	*Laboratory session*				
Tue	Lecture	13.15 - 15.00	*HC3*	Scheduling: static and dynamic priorities, utilization bound analysis	notes
Wed	*Laboratory sessions*				
Thu	Lecture	13.15 - 15.00	*HC3*	Scheduling: response time analysis	notes
Thu	Exercise	15.15 - 17.00	*HC3*	Uniprocessor schedulability analysis	notes
Fri	Lecture	13.15 - 15.00	*HC3*	Scheduling: processor demand analysis	notes
Week 7					**Feb 26 - Mar 2**
Mon	*Laboratory session*				
Tue	*Laboratory session*				
Tue	Exercise	13.15 - 15.00	*HC3*	Uniprocessor schedulability analysis	notes
Wed	*Laboratory sessions*				
Thu	Lecture	13.15 - 15.00	*HC3*	Scheduling: multiprocessor systems	notes
Thu	Exercise	15.15 - 17.00	*HC3*	Multiprocessor schedulability analysis	notes
Fri	*Special*	13.15 - 15.00	*HC3*	Insights on scheduling; old exam problems	
Week 8					**Mar 5 - Mar 9**
Mon	*Laboratory session*				
Tue	*Laboratory session*				
Tue	Lecture	13.15 - 15.00	*HC3*	Summary and reading hints; old exam problems	notes
Wed	*Laboratory sessions*				
Thu	*Special*	15.15 - 17.00	*HA3*	Old exam problems	
Fri					

CHALMERS | UNIVERSITY OF GOTHENBURG

Course aim

After the course, the student should be able to:

- Formulate requirements for embedded systems with strict constraints on computational delay and periodicity.
- Categorize and describe the different layers in a system architecture for embedded real-time systems.
- Construct concurrently-executing tasks for real-time applications that interface to hardware devices (sensors/actuators)
- Describe the principles and mechanisms used for designing run-time systems and networks for real-time applications.
- Apply the basic analysis methods used for verifying the temporal correctness of a set of executing tasks.

CHALMERS | UNIVERSITY OF GOTHENBURG

Course examination

Written exam

The course contents are examined by means of a written exam.
Date and time of ordinary exam: March 12, 2018 @ 08:30-12:30.

Laboratory assignment

The laboratory assignment is examined by means of:
(i) parts 0, 1 and 2 of the Lab-PM, and (ii) a project report.

Final grade

The written exam and the laboratory assignment are both given a score with grade (U, 3 – 5). To pass the course the score of each much be equivalent to a grade of 3 or higher. The final grade is based on the scores from the exam and the assignment.

CHALMERS | UNIVERSITY OF GOTHENBURG

Course material

To download: (via course web page)

- Lecture notes and exercise notes.
- *Programming with the TinyTimber kernel.* [also used at written exam]
- Research articles and book excerpts. [recommended reading only]
- Exercise compendium.
- Lab-PM – Part 0, 1 and 2. [paper copies also handed out]
- Template code. [for target computer software]
- Handbooks and data sheets. [for target computer hardware]
- Development tools. [available for Windows, Mac, Linux]

CHALMERS | UNIVERSITY OF GOTHENBURG

Course information and support

Teachers and assistants:
- Primarily in conjunction with lectures and exercise/lab sessions.
- Otherwise send an email or book time for a personal meeting.

PingPong:
- Laboratory assignment: form groups and submit reports
- Results regarding laboratory assignment and written exam

Course web page:
- To get complete information about the course
- To download course material

Follow @rtsyschalmers

`http://www.cse.chalmers.se/edu/course/EDA223`

CHALMERS | UNIVERSITY OF GOTHENBURG

Course contents

What this course is all about:

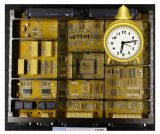
1. Construction methods for real-time systems
   - Specification, implementation, verification
   - Application constraints: origin and implications
2. Programming of concurrent real-time applications
   - Task and communication models (C with TinyTimber kernel)
   - I/O and interrupt programming (C with TinyTimber kernel)
3. Verification of system's temporal correctness
   - Fundamental scheduling theory
   - Derivation of worst-case task execution times

CHALMERS | UNIVERSITY OF GOTHENBURG

What is a real-time system?

> "A real-time system is one in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are generated"

J. Stankovic, "Misconceptions of Real-Time Computing", 1988



CHALMERS | UNIVERSITY OF GOTHENBURG

What is a real-time system?

It is <u>not only</u> about high-performance computing!

Real-time systems <u>must meet</u> timing constraints

High-performance computing <u>maximizes average</u> throughput

Average performance says nothing about correctness!

"A statistician drowned while crossing a stream that was, on average, 6 inches deep"

Real-time system are instead usually optimized with respect to perceived "robustness" (control systems) or "comfort" (multimedia)

What is a real-time system?

Typical properties of a real-time system:

- Application-specific design
  - Part of a bigger system ("embedded system")
  - Carefully specified system properties
  - Well-known operating environment
- Strict timing constraints
  - Responsiveness (= deadline)
  - Periodicity (= sampling rate)
  - Important design parameters in the context of system safety
  - Should be verified at design time to be met at run-time

CHALMERS | UNIVERSITY OF GOTHENBURG

What is a real-time system?

Typical properties of a real-time system (cont'd):

- Strict safety requirements
  "Safety must be considered early in the design process"
  - Safety standards and certification
    - IEC 61508 (industrial systems)
    - IEC 62304 (medical systems)
    - ISO 26262 (automotive systems)
    - DO-178C (airborne systems)
  - Programming language restrictions (e.g. MISRA C)
  - Run-time system restrictions (e.g. cyclic executive)
  - Thorough testing of software and hardware components
  - Reliability in presence of component faults ("fault tolerance")

CHALMERS | UNIVERSITY OF GOTHENBURG

What is a real-time system?

Examples of real-time systems:

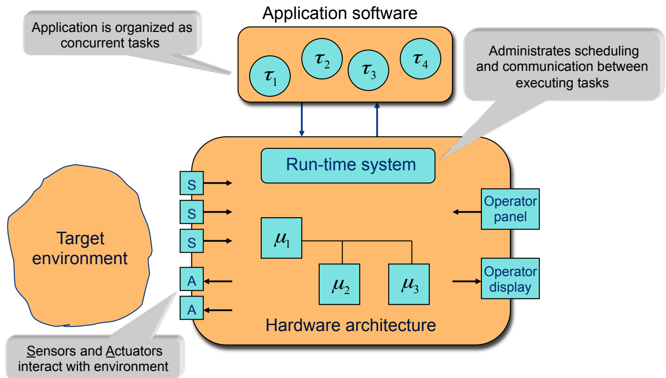
- Control systems
  - Manufacturing systems; process industry
  - Cars, aircrafts, submarines, satellites

- Transaction systems
  - E-commerce; ticket booking; teller machines; stock exchange
  - Wireless phones; telephone switches

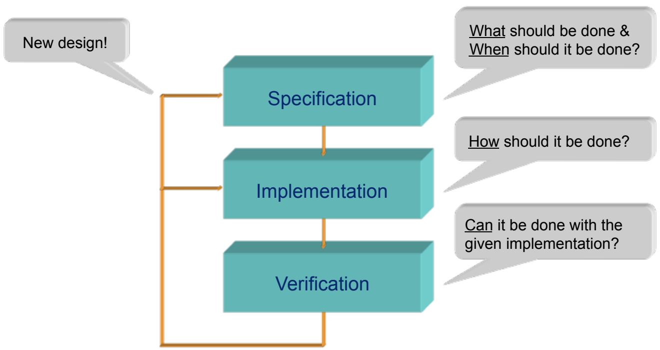
- Multimedia
  - Portable music players, streaming music
  - Computer games; video-on-demand, virtual reality

CHALMERS | UNIVERSITY OF GOTHENBURG

Real-time system components



Designing a real-time system



Specification

Specification → Implementation

Requirements:		Constraints:
Reliability	→	Replication
Sampling rate	→	Periodicity
Response time	→	Deadline
Resources	→	Locality

Specification

Examples of application constraints:

- Timing constraints
  - A task must complete its execution within given time frames (example: task periodicity or deadline)
- Exclusion constraints
  - A task must execute a code region without being interrupted (example: a task needs exclusive access to a shared resource)
- Precedence constraints
  - A task must complete its execution before another task can start (example: a data exchange must take place between the tasks)

Specification

Where do the timing constraints come from?

- Laws of nature
  - Bodies in motion: arm movements in a robotic system
  - Inertia of the eye: minimal frame rate in film
- Mathematical theory
  - Control theory: recommended sampling rate
- Component limitations
  - Sensors and actuators: minimal time between operations
- Artificial derivation
  - Observable events: overall (global) timing constraints are given, but individual (local) timing constraints are needed

Specification

How critical are the constraints?

Hard constraints:

If the system fails to fulfill a timing constraint, the computational results is useless.

Non-critical: system can still function with reduced performance
- Navigational functions; diagnostics

Critical: system cannot continue to function
- Flight control system; control loop

Safety-critical: can cause serious damage or even loss of life
- Braking systems (ABS); defense system (missiles, robots)

Correctness must be verified before system is put in mission!

CHALMERS | UNIVERSITY OF GOTHENBURG

Specification

How critical are the constraints?

Soft constraints:

Single failures to fulfill a timing constraint is acceptable, but the usefulness of the computational result is reduced (often to what can be considered useless).

- Reservation systems: seat booking for aircraft; teller machine
- E-commerce: stock trading, eBay
- Multimedia: video-on-demand, computer games, Virtual Reality

Statistical guarantees often suffice for these systems!

CHALMERS | UNIVERSITY OF GOTHENBURG

Implementation

Critical choices to be made at design time:

- Application software
  - Programming language: determines run-time performance, code size and degree of timing verification that is possible.
  - Concurrency: determines degree of application parallelism that can potentially be exploited by the hardware.
- Run-time system:
  - Task and message scheduling policy: determines potential of meeting timing constraints, and sets limits of maximum processor and network utilization.

CHALMERS | UNIVERSITY OF GOTHENBURG

Implementation

Critical choices to be made at design time:

- Hardware architecture:
  - Hardware parallelism: determines the degree of application parallelism that can actually be exploited
    - uniprocessor system: only pseudo-parallel execution is possible
    - multiprocessor system: truly parallel execution is possible
  - Microprocessor family: determines run-time performance, as well as difficulty in analyzing the worst-case execution time (WCET) of a software task.
  - Communication network technology: determines run-time performance, as well as difficulty in analyzing worst-case message delays.

CHALMERS | UNIVERSITY OF GOTHENBURG

Verification

How do we verify the system?

Ad hoc testing:

Run the system for ”a while” and let the absence of failures ”prove” the correctness

- fast method that indicates that ”everything seems to work”
- pathological cases can be overlooked during testing
- too frequently used as the only method in industrial design

Exhaustive testing:

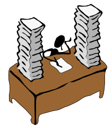
Verify all combinations of input data, time and faults

- considers all possible cases
- requires an unreasonable amount of time for testing

CHALMERS | UNIVERSITY OF GOTHENBURG

Verification

How do we verify the system?

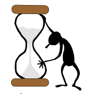


Formal analysis of the implementation:

Verify logical correctness using proof machine

- requires dedicated description language
- abstraction level very high (often implementation independent)

Verify temporal correctness using schedulability analysis



- necessary for verifying hard-real-time systems
- requires WCET for each task
- requires support in programming language and run-time system

Note: results from the verification phase are only valid if all assumptions actually apply at run-time!

CHALMERS | UNIVERSITY OF GOTHENBURG

Verification

What sources of uncertainty exist in formal verification?

- Non-determinism in tasks’ WCET (undisturbed execution)
  - Input data and internal state controls execution paths
  - Memory access patterns control delays in processor architecture (pipelines and cache memories)
- Non-determinism in tasks’ execution interference
  - Run-time system’s scheduling policy controls interference pattern of tasks with pseudo-parallel execution
- Conflicts in tasks’ demands for shared resources
  - Pseudo-parallel task execution may give rise to uncontrolled blocking of shared hardware and software resources