Embedded Systems

Bernd Finkbeiner
Calogero Zarba

Sommersemester 2007

Course Info: Embedded Systems

- Stammvorlesung Praktische Informatik
- 9 LP
- Tuesdays, Thursdays 9-11
  Building E 1 3, HS 003
- http://react.cs.uni-sb.de/courses/es
- The team:
  - Bernd Finkbeiner
    building 45, office 506
    phone (0681) 302-5632
    finkbeiner@cs.uni-sb.de
  - Calogero Zarba
    building 45, office 507
    phone (0681) 302-5622
    zarba@cs.uni-sb.de
  - Moritz Hahn
    emh@studcs.uni-sb.de

This course is about…

Embedded Systems

Estimates for number of embedded systems in current use: >10^{10}

[Rammig 2000, Motorola 2001]

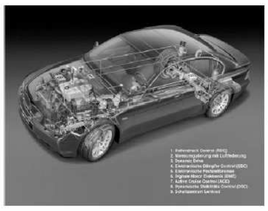
400 horses

100 microprocessors

Automotive SW Workshop
San Diego, USA
H.-G. Frischkorn
BMW Group
10 -12. Jan. 2004
Page 4

Automotive Software: An Emerging Domain
A Software Perspective

- Up to 40% of the vehicles' costs are determined by electronics and software
- 90% of all innovations are driven by electronics and software
- 50 – 70% of the development costs for an ECU are related to software
- Premium cars have up to 70 ECUs, connected by 5 system busses

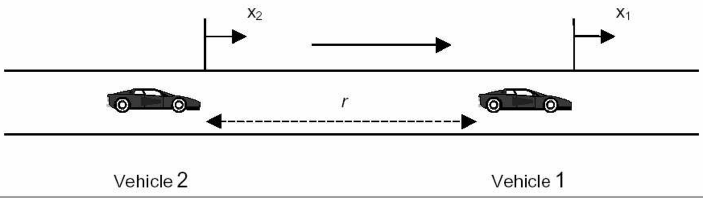


- **Growing system complexity**
- **More dependencies**
- **Costs play significant role**

Intelligent Cruise Control

Cooperative Adaptive (Intelligent) Cruise Control (CACC):

Cruise at given speed when the road is clear (cruise control) , otherwise follow the car in front, using radar (adaptive) and/or communications (cooperative).

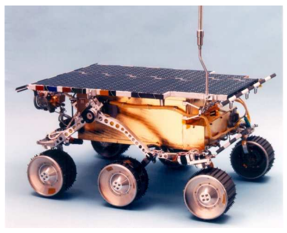


Thanks to PATH publication unit

Mars, July 4, 1997

The MARS Pathfinder problem

“But a few days into the mission, not long after Pathfinder started gathering meteorological data, the spacecraft began experiencing total system resets, each resulting in losses of data. The press reported these failures in terms such as "software glitches" and "the computer was trying to do too many things at once".” …



The MARS Pathfinder problem

- System overview:
  - ***Information Bus (IB):***
    - Buffer for exchanging data between different tasks
    - Shared resource of two tasks M and B
  - Three tasks:
    - ***Meteorological data gathering task (M):***
      - collects meteorological data
      - reserves IB, writes data to IB, releases IB
      - infrequent task, low priority
    - ***Bus management (B):***
      - data transport from IB to destination
      - reserves IB, data transport, releases IB
      - frequent task, high priority

The MARS Pathfinder problem

- Three tasks:
  - ***...***
  - ***“Communication task” (C):***
    - medium priority, does not use IB
- Scheduling with fixed priorities.
- ***Watch dog timer (W):***
  - Execution of B as indicator of system hang-up
  - If B is not activated for certain amount of time: Reset the system

The MARS Pathfinder problem

(see http://research.microsoft.com/~mbj/Mars_Pathfinder/)

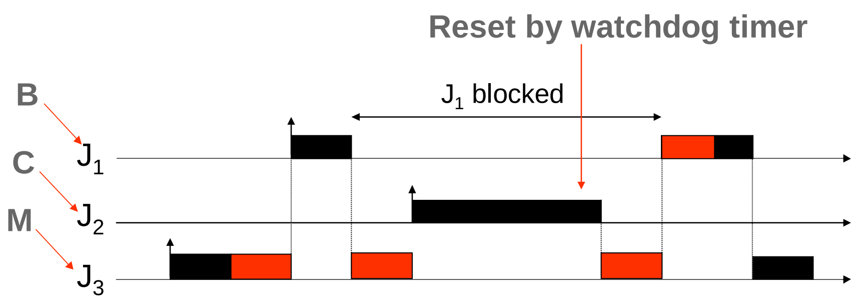
“Most of the time this combination worked fine.

However, very infrequently it was possible for an interrupt to occur that caused the (medium priority) communications task to be scheduled during the short interval while the (high priority) information bus thread was blocked waiting for the (low priority) meteorological data thread. In this case, the long-running communications task, having higher priority than the meteorological task, would prevent it from running, consequently preventing the blocked information bus task from running.

After some time had passed, a watchdog timer would go off, notice that the data bus task had not been executed for some time, conclude that something had gone drastically wrong, and initiate a total system reset.

This scenario is a classic case of priority inversion.”

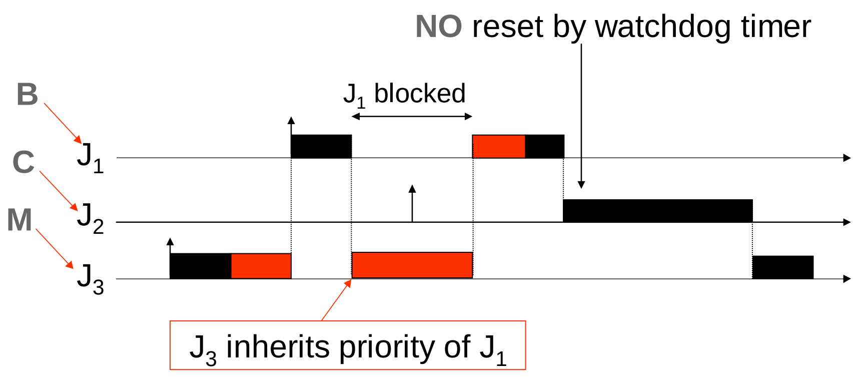
Priority inversion



■ normal execution 🟥 critical region

priority(J_1) > priority(J_2) > priority(J_3)

Classic solution: Priority inheritance



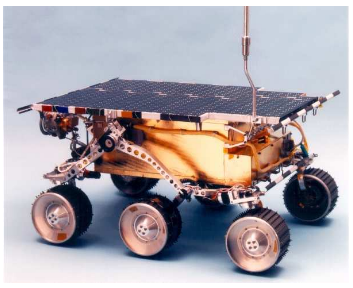
■ normal execution 🟥 critical region

priority(J_1) > priority(J_2) > priority(J_3)

Priority inversion on Mars

- Priority inheritance also solved the Mars Pathfinder problem:
  - the VxWorks operating system used in the pathfinder implements a flag for the calls to mutual exclusion primitives.
  - This flag allows priority inheritance to be set to “on”.
  - When the software was shipped, it was set to “off”.

The problem on Mars was corrected by using the debugging facilities of VxWorks to change the flag to “on”, while the Pathfinder was already on the Mars [Jones, 1997].



Tutorial

Tutorial A: Wednesdays 11-13	Room 016 E 1 3
Tutorial B: Fridays 14-16	Room 015 E 1 3

Starting next week
Please sign up for one of the two tutorials.

There will be weekly homeworks
distributed on Tuesday (first one: **today**) and
due on the following ~~Mondays (10:59am)~~ **Tuesdays in the lecture**

„Paper & Pencil“ homeworks + Exercises with Matlab/Stateflow

Exams

- Midterm: May 22, 2007
- Final: July 26, 2007
- Backup Exam: September 28, 2007

Requirement for admission to final exam:

- > 50% of points in homeworks

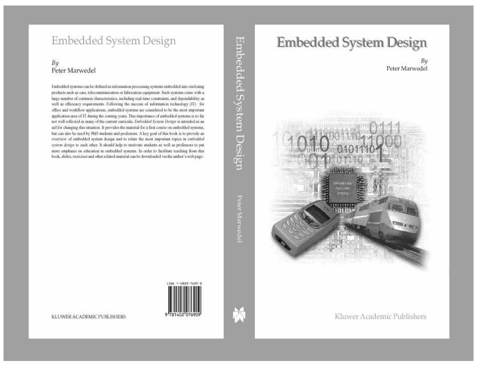
Requirement for admission to backup exam:

- > 50% of points in homeworks, and
- passing grade in midterm or final, but not in both

You pass the course if you pass two exams.

Your final grade is the average of the two passing grades.

Textbook



Peter Marwedel.
Embedded System Design.
Springer, Berlin;
2nd Print (1. November 2005)
ISBN-10: 0387292373

Other Recommended Literature

HARD REAL-TIME COMPUTING SYSTEMS
Predictable Scheduling Algorithms and Applications
Second Edition
Giorgio C. Buttazzo
Springer

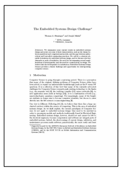
Giorgio C. Buttazzo
Hard Real-Time Computing Systems

Jürgen Teich,
Digitale Hardware/Software Systeme

Heinz Wörn, Uwe Brinkschulte,
Echtzeitsysteme

Introduction

The Embedded Systems Design Challenge



Recommendation: „*The Embedded Systems Design Challenge*“ by Henzinger and Sifakis

See link on webpage

Embedded Systems

Embedded system = **engineering artifact involving computation that is subject to physical constraints**

Constraint #1: Reaction to the physical environment

Reaction constraints: deadlines, throughput, jitter

Constraint #2: Execution on a physical platform

Execution constraints: Bounds on available processor speeds, power, hardware failure rates

Challenge: Gain control over the interplay of computation with reaction and execution constraints, so as to meet given requirements.

Characteristics of Embedded Systems

Must be **dependable**:

Reliability *R(t)* = probability of system working correctly provided that is was working at *t*=0

Maintainability *M(d)* = probability of system working correctly *d* time units after error occurred.

Availability *A*(*t*): probability of system working at time *t*

Safety: no harm to be caused

Security: confidential and authentic communication

Even perfectly designed systems can fail if the assumptions about the workload and possible errors turn out to be wrong.

Making the system dependable must not be an after-thought, it must be considered from the very beginning.

Characteristics of Embedded Systems

Must be **efficient:**

- **Energy** efficient
- **Code-size** efficient (especially for systems on a chip)
- **Run-time** efficient
- **Weight** efficient
- **Cost** efficient

Dedicated towards a certain application
Knowledge about behavior at design time can be used to minimize resources and to maximize robustness

Dedicated user interface
(no mouse, keyboard and screen)

Characteristics of Embedded Systems

Many ES must meet real-time constraints

A real-time system must react to stimuli from the controlled object (or the operator) within the time interval dictated by the environment.

For real-time systems, right answers arriving too late are wrong.

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

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

Characteristics of Embedded Systems

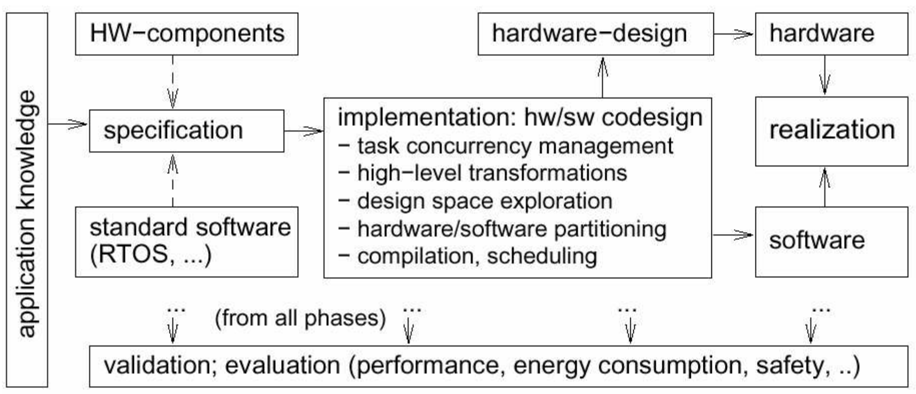
Frequently connected to physical environment through sensors and actuators.

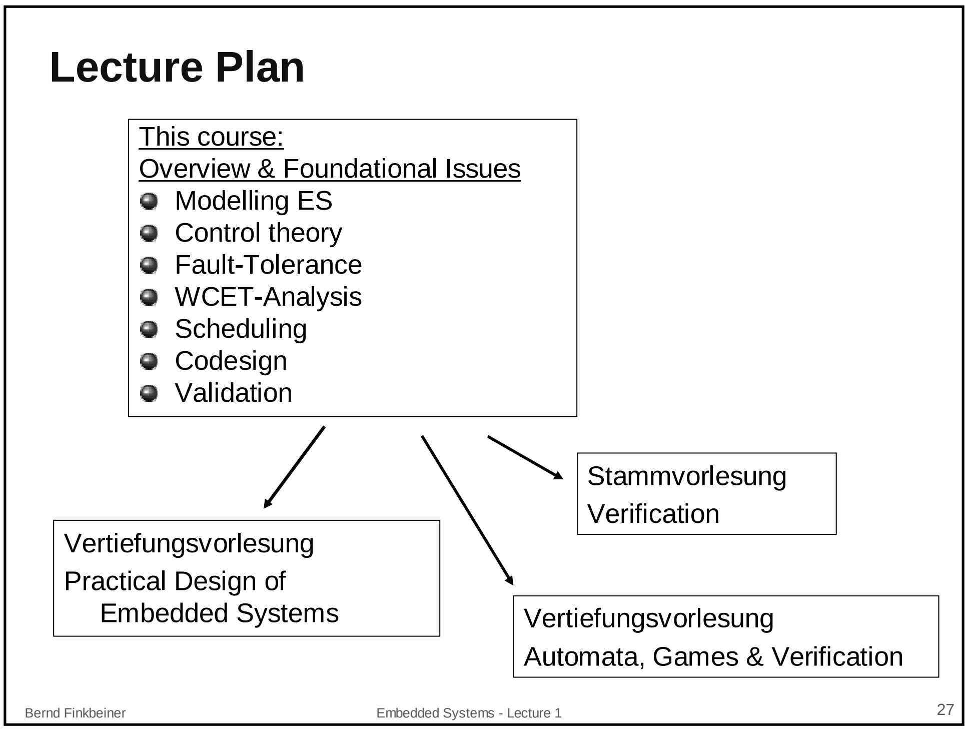
Typically Embedded Systems are

- **Hybrid systems** (analog + digital parts)
- **Reactive systems**

„A reactive system is one which is in continual interaction with is environment and executes at a pace determined by that environment“ [Bergé, 1995]
Behavior depends on input and current state.

Design Flow According to Marwedel‘s book


application knowledge
HW−components
specification
standard software (RTOS, ...)
hardware−design
hardware
implementation: hw/sw codesign
− task concurrency management
− high−level transformations
− design space exploration
− hardware/software partitioning
− compilation, scheduling
realization
software
(from all phases)
validation; evaluation (performance, energy consumption, safety, ..)



Lecture Plan

<u>This course:</u>
<u>Overview & Foundational Issues</u>

- Modelling ES
- Control theory
- Fault-Tolerance
- WCET-Analysis
- Scheduling
- Codesign
- Validation

Stammvorlesung
Verification

Vertiefungsvorlesung
Practical Design of
Embedded Systems

Vertiefungsvorlesung
Automata, Games & Verification

Bernd Finkbeiner Embedded Systems - Lecture 1 27