Embedded Systems Development

Lecture 1
Introduction

Daniel Kästner
AbsInt Angewandte Informatik GmbH
kaestner@absint.com
Overview

- Daniel Kästner, Florian Martin, Marc Schlickling.
- Advanced course (6CP): Fr 14-16, E1.3, HS003. 2 hours exercise.
  Goal: Working with industry tools for embedded systems development and understanding their theoretical background.
- Contents: Model-based code generation, task scheduling and schedulability analysis, worst-case execution time analysis, code generation for embedded processors.
- Tools used:
  - SCADE: CASE tool for safety-critical embedded systems (avionics).
  - Symta/S: Task scheduling & schedulability analysis (automotive).
  - aiT WCET Analyzer: Worst-case execution time analysis (avionics & automotive).
  - StackAnalyzer: Worst-case stack consumption (avionics & automotive).
- Practical project with LEGO Mindstorms.
Organization

- Website:  [http://rw4.cs.uni-sb.de/teaching/esd07](http://rw4.cs.uni-sb.de/teaching/esd07)
- Mailing Lists:
  - esd07@gigasun.cs.uni-sb.de
  - esd07-tutor@gigasun.cs.uni-sb.de
- Exercises
  - Teams of 1-2 people allowed, but have to be announced.
  - First tutorial in week from 12.11.-15.11.
  - Potential tutorial dates: Mi 10-12, or Do 18-20.
- Written examination: 15.02.2008, E1.3, HS003.
  - at least 40% of total exercises points
  - at least 10% of each week’s assignment
  - successful participation in project
  - Final grade composed from examination result and potential bonus points from project.
- Project phase:
  - Start 11.01.
Motivation

“Information technology (IT) is on the verge of another revolution. Driven by the increasing capabilities and ever declining costs of computing and communications devices, IT is being embedded into a growing range of physical devices linked together through networks and will become ever more pervasive as the component technologies become smaller, faster, and cheaper... These networked systems of embedded computers ... have the potential to change radically the way people interact with their environment by linking together a range of devices and sensors that will allow information to be collected, shared, and processed in unprecedented ways. ... The use of [these embedded computers] throughout society could well dwarf previous milestones in the information revolution.”

National Research Council Report (US) Embedded Everywhere


Courtesy: P. Marwedel
More Motivation

- Excellence cluster in Saarbrücken: Multi-modal Computing and Interaction
- Goal: develop computing systems that can interact with humans in a natural way. They should be
  - able to process different kinds of information: speech, images, videos, graphics, ...
  - pervasive: be available anytime, anywhere
  - reactive: analyze their environment, react to speech, text, gestures.
- Embedded systems all over.
- Selected challenges: distributed systems, real-time processing, safety.

[Source: Press release of Saarland University Computer Science Dptmt]
Application Areas of Embedded Systems

- **Automotive**
  - Up to 100 microprocessors per car.
  - Networked together
  - Applications
    - Engine control
    - Active suspension
    - Air-conditioning
    - Airbag
    - Navigation systems (GPS)
    - Sound system, active noise cancellation
    - Blind-angle alert systems
    - ...
Application Areas of Embedded Systems

- Avionics
  - Pilot information systems
  - Breaking & steering system
  - Security control
  - Anti-collision systems
  - Fly-by-wire
  - Remote Piloted Vehicles

- Space
  - Autonomous vehicles
  - Satellite control
  - ...
Application Areas of Embedded Systems

- Consumer electronics
  - AV-R Receivers (e.g. Analog Devices SHARC)
  - Smart Pen
  - CD-player, DVD-player, MP3-player
  - Organizer, PDAs
  - Washing machines, microwave ovens, ...
  - PC peripherals (hard disk control, graphics cards, ...)
Application Areas of Embedded Systems

- Telecommunications
  - Telecom switch
  - Cell phones
  - ISDN phones, fax
  - Answering machines, etc

- DSP-Applications:
  - voice and data compression
  - echo reduction
  - signal multiplexing
  - filtering
Application Areas of Embedded Systems

- Military:
  - Radar
  - Navigation
  - Weapons
  - Damage control

- Healthcare Technologies
  - Diagnostic imaging (Computed Tomography, Magnetic Resonance Imaging, ultrasound, etc)
  - Electrocardiagram analysis
  - Medical image storage/retrieval
  - Artificial eye
Definitions

- **System**: A system is a portion of the universe that has been chosen for studying the changes that take place within it in response to varying conditions [Encyclopedia Britannica].

- **Model**: Any real situation in the physical or biological worlds is subject to analysis by modelling if it can be described in terms of mathematical equations. As such, a model is a simplified representation of the real world including only those variables relevant to the problem at hand [Encyclopedia Britannica].
Classifications of Systems

- **System categories:**
  - sequential vs parallel,
  - central vs distributed,
  - deterministic vs nondeterministic,
  - terminating vs nonterminating

- **System types:**
  - **transformational:** input/output systems, computations terminate.
  - **interactive:** continuous interaction with environment, computer controls the pace, not necessarily deterministic.
  - **reactive:** continuous interaction with environment, environment controls the pace, generally deterministic, timing of inputs and outputs is important, often critical reliability requirements.
Embedded Systems: Definition

- **Embedded systems** are
  - embedded in a **physical environment** and **interact** with it for measuring or controlling purposes.
  - Information processing systems embedded into a larger product; main reason for buying is **not** information processing [Marwedel].

- **Characteristics of embedded systems:**
  - **complex interaction** with environment
  - usually **dedicated** towards a certain application
  - typically **reactive systems**
  - often **safety-critical**
  - often **real-time** processing required
Special Case: Real-Time Systems

- In a **real-time system**, the correctness not only depends on the logical results but also on the **timing** of the applications.

- **Distinction:**
  - **Hard** real-time system: It is **vital** that the system satisfies the timing condition. Failure results in **catastrophic** consequences, e.g. the loss of lifes. Examples: flight control software, airbag control.
  - **Soft** real-time system: It is **desirable** that the system satisfies the timing conditions; otherwise the functioning of the system is negatively affected. Example: MP3-Player, telephone software.
Characteristics of Embedded Systems (c’ed)

- **High dependability** requirements:
  - Reliability $R(t) = \text{probability of system working correctly provided that it was working at } t=0$.
  - Maintainability $M(d) = \text{probability of system working correctly } d \text{ time units after error occurred}$.
  - Availability $A(t): \text{probability of system working at time } t$.
  - 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 (c’ed)

- Must be **efficient**
  - energy efficient
  - code-size efficient
  - runtime efficient
  - weight efficient
  - cost efficient

- Trend to **replace** conventional mechanics by digital embedded components. Reasons:
  - production cost,
  - functionality,
  - weight,
  - size.
Market for Embedded Systems

- **Embedded chips** account for more than 90% of all silicon processors sold [Leibson(Embedded Processor Forum), 2001]

- Market for **embedded processors** estimated at almost $50 billion ($10^9) in 1997; annual growth estimated at 35% per year [Micrologic Research].

- Worldwide **mobile phone sales** surpassed 156.4 mln units in Q2 2004, a 35% increase from Q2 2003, according to Gartner [www.itfacts.biz]

- The worldwide **portable flash player** market exploded in 2003 and is expected to grow from 12.5 mln units in 2003 to over 50 mln units in 2008 [www.itfacts.biz]
Market for Embedded Systems (2)

- Today’s DVR (digital video recorders) users - 5% of households - will grow to 41% within five years, according to Forrester. [www.itfacts.biz]

- The automotive sector … ensures the employment of more than 4 million people in Europe. Altogether, some 8 million jobs in total depend on the fortunes of the transport industry and related sectors - representing around 7% of the European Union’s Gross National Product (GNP) [OMI bulletin]

- .. but embedded chips form the backbone of the electronics driven world in which we live … they are part of almost everything that runs on electricity [Mary Ryan, EEDesign, 1995]
Challenges

- **Dependability?**
  - Non-real time protocols used for real-time applications
  - Over-simplification of models (e.g. aircraft anti-collision system)
  - Using unsafe systems for safety-critical missions (e.g. voice control system in LA; ~800 planes without voice connection to tower for >3hrs)

- **Challenges for embedded software**
  - Dynamic environments
  - Capture the required behaviour!
  - Validate specifications
  - Efficient translation of specifications into implementations!
  - How can we check that we meet real-time constraints?
  - How do we validate embedded real-time software? (large volumes of data, testing may be safety-critical)
Software Development

- Waterfall model
  - Classic software life cycle model; until early 1980s the only widely accepted life cycle model Schach [1999].
  - Represents the software life cycle using processes and products.
  - Each process transforms a product to produce a new product as output. Then the new product becomes the input of the next process.
  - Important characteristics: processes are iterative.

- V-Model
  - Regulates “who”, “when”, “what” in a software development project.
  - Development standard for IT systems of the German Federation for the entire civil and military area.
  - Basics: hierarchical decomposition of system into smaller parts until realization becomes possible.
  - Verification and validation is done on each construction stage.
  - No strict temporal ordering imposed.
Software Development

- **Waterfall model**
  - Classic software life cycle model; until early 1980s the only widely accepted life cycle model [Schach, 1999].
  - Represents the software life cycle using processes and products.
  - Each process transforms a product to produce a new product as output. Then the new product becomes the input of the next process.
  - Important characteristics: processes are iterative.

- **V-Model**
  - Regulates “who”, “when”, “what” in a software development project.
  - Development standard for IT systems of the German Federation for the entire civil and military area.
  - Basics: hierarchical decomposition of system into smaller parts until realization becomes possible.
  - Verification and validation is done on each construction stage.
  - No strict temporal ordering imposed.
V-Model

Requirements Specification
System Specification
System Validation
Installation Acceptance Test
Architectural Design
System Integration & Test
Subsystem Design
Subsystem Integration & Test
Module Design
Module Testing
Coding
System Integration & Test
Module Testing
Coding
System Validation
Installation Acceptance Test
Architectural Design
System Specification
Requirements Specification
Development of Avionics Software

- **DO-178B** Standard: guidelines for the production of software for airborne systems and equipment.
  - Development assurance levels:
    - **A**: catastrophic failure condition for the aircraft (e.g. aircraft crash)
    - **B**: Hazardous/severe failure condition for the aircraft (e.g. injured persons)
    - **C**: Major failure condition for the aircraft (e.g. flight management system down => manual operation by pilot)
    - **D**: Minor failure condition for aircraft (e.g. pilot-ground communications down)
    - **E**: No effect on aircraft operation or pilot workload (e.g. entertainment system down)
DO-178B Standard

- **Essence:** formulation of appropriate *objectives/requirements* and verification that these objectives have been achieved. The ways of achieving an objective may vary.

- **Purpose:** detect and report errors that may have been introduced during the software development process.

- **Important:** All requirements have to be *verifiable* and must be *compliant* with the requirements of other stages.

- **Testing** is part of the verification process, but *reviews* and *analyses* are also required. Analyses should be reproducible.
DO-178B Development Process

System Requirements Process

System Requirements allocated to Software

SW Requirements Process

High-level requirements

SW Design Process

Low-level requirements & architecture

Change requests

SW Coding Process

Source Code

Change requests

SW Integration Process

Integrated Executable

Software Development Processes (DO-178B)

System Development Processes (ARP4754)
DO-178B Verification Process

- Reviews and Analyses of the High-Level Requirements:
  - Algorithm accuracy

- Reviews and Analyses of the Low-Level Requirements:
  - Compatibility with target computer: no conflict between software requirements and hardware/software features of the target computer, e.g. system response times, input/output hardware

- Reviews and analyses of the source code:
  - Verifiability: the source code does not contain statements and structures that cannot be verified and the code does not have to be altered to test it.
  - Accuracy and consistency: stack usage, resource contention, worst-case execution timing, exception handling, use of non-initialized variables or constants.
Development of Avionics Software

- Airbus A340 contains 115 digital units and 20 MB onboard software.

- Development of safety-critical avionics software is very expensive:
  - Avg development and test of 10 KLOC level B software is 16 person-years
  - Cost of minor bug is $100K-$500K
  - Cost of major bug is $1M-$500M
  - Time-to-market 3-4 years
  - For Level A software, the overall verification cost (including testing) may account for up to 80% of the budget
Why so expensive?

- Multiple descriptions:
  - At each level software is *rewritten* into another form – traditionally by hand => expensive and error-prone.

- **Ambiguity and lack of accuracy** of specifications.

- **Manual** coding

- **Late detection** of specification and design errors
Model-based Development

- Model is software specification.

- Hardware/Software codesign.
- Prototyping.
- Formal verification.

- Automated & integrated development methods and tools:
  - Simulation.
  - Documentation.
  - Automatic code generation.

- Automated & integrated analysis and test methods
  - Static system analysis
  - Synthesis of test suites
Model-based Software Development

Lustre programs
Esterel programs

Generator

Compiler

C Code

Esterel SCADE
- SCADE language
- SyncCharts

aiT WCET Analyzer
- Timing Validation

SymTA/S
- System-level Schedulability Analysis

Compiler

Binary Code
Lego Mindstorms NXT

- A brief overview of the NXT hardware based on information from LEGO. The LEGO Mindstorms NXT consists of following items:
  - **NXT brick**
    - CPU: 32-bit ARM7 micro controller @ 48 MHz
    - Co-Processor: 8-bit AVR micro controller @ 4 MHz
    - 4 input ports / 3 output ports
    - 100 x 64 pixel LCD display
    - USB 2.0 and Bluetooth support
    - Speaker
  - **Sensors**
    - Ultrasonic sensor
    - Touch sensor
    - Sound sensor
    - Light sensor
  - **Motors**
    - 3 motors with integrated rotation sensors
Mindstorms Project Platform

- **LEJ OS OSEK**
  - LEJ OS NXJ - Java for Lego Mindstorms ("tiny" Java Virtual Machine)
  - TOPPERS OSEK - Toyohashi OPeN Platform for Embedded Real-time Systems

- **Provides**
  - ANSI C/C++ programming environment using GCC tool chain
  - LEJ OS NXJ platform based C API for NXT Sensors, Motors, and other devices
  - (TOPPERS) OSEK provided real-time multi tasking features proven in automotive industry
Contents of Lecture

- Basic Automata Theory
- SyncCharts / Safe State Machines (SSM)
- Synchronous languages: Esterel and Lustre.
- The SCADE textual and graphical languages
- Compilation for Embedded Processors
- Static Program Analysis: principles, stack analysis, worst-case execution time analysis.
- Scheduling & Schedulability Analysis