Embedded Control System Implementation: Method and Co-Simulation

Jan F. Broenink, Pieter Maljaars
with contributions of:
Marcel Groothuis, Peter Visser, Bojan Orlic, Dusko Jovanovic, Gerald Hilderink

Control Engineering, CTIT, Faculty EE-M-CS,
University of Twente, Enschede, Netherlands

Overview 3rd lecture: Design Method

- Method & …
  - 4 steps revisited and detailed
  - Focus on Embedded Software -> exercise
- … Techniques
  - Co-simulation
  - Hardware-in-the-Loop Simulation
- Cases
  - On Co-simulation
  - On Hardware-in-the-Loop Simulation
- Overview

Layered Structure of Controllers

- Timing
  - Hard real time
    Safety, Loop Controllers, Sequence Controllers (set point generators)
  - Soft real time
    Sequence Controllers, Supervisory Controller, User Interface

gCSP – Course Overview

- Introduction & Context
  - Setting the scene: embedded control systems software
  - Showcase: Production Cell
- CSP and gCSP
  - Basics of the process algebra – practical point of view
  - gCSP the tool / techniques
- Exercises I
  - gCSP, basics, simple control example
- Method & CTC++
  - Design method
  - Underlying execution framework
  - Linkdrivers, timers
- Exercises II
  - Control a robot
- Wrap up

Lunch break
Model-Based Embedded Control Software Development

Jan F. Broenink, Pieter Maljaars
with contributions of: Marcel A. Groothuis, Peter M. Visser,
Bojan Orlic, Dusko Jovanovic, Gerald Hildierink

### Design Approach
- **Integrated Models**
  - Discipline-specific models connected
  - Total system (embedded + embedding!)
- **Virtual Prototyping = Simulation**
  - Layered structure of controllers
  - Continuous time & discrete event
- **Stepwise Refinement**
  - Physical Systems modeling
  - Control law Design
  - Embedded Control System Implementation
    » Gradually enhance laws to code
    » Realization (software)
- **Tools needed**
  - Extendable / updatable models & software
  - Total system (embedded + embedding!)

### Embedded Control System Implementation – Approach
- **Design method for ECS (software)**
  - Stepwise Refinement
    » Intrinsically iterative
  - Verification via
    » Simulation / Co-Sim. / X-in-the-Loop Sim. (hardware / software)
    » Formal verification: model checking on deadlocks
  - Design Space Exploration
    » To support decisions on design choices
    » Concurrent engineering / Mechatronic approach
    » “Doing first time right” -> as little test time on target machine as possible
- **Stepwise Refinement**
  - Physical Systems modeling
  - Control law Design
  - Embedded Control Software Implementation
    » Gradually enhance laws to code
    » Realization (software)
- **Tools needed**
  - Extendable / updatable models & software
  - Total system (embedded + embedding!)

### Physical Systems Modeling / Control Law Design
- **Physical Systems Modeling**
  - Goals of Competent Model
    » Understand dynamics
    » Derive control laws
    » Test system
  - Generate detailed model
    » Library blocks
    » Verify model by simulation
    » Model parts
    » Tests suitable for validation
    » Validate by measurements
- **Control Law Design**
  - Simplified Model
    » Model reduction
    » Model linearization
  - Verify Simplified Model
    » Compare to detailed model
    » Derive Control Law(s)
  - Use simplified model
    » Use detailed model
    » Combine Control Laws

### ECS implementation Design Trajectory
- **Physical System Modeling**
  - Plant Design
  - Detailed model
  - Library blocks
  - Model parts
- **Control Law Design**
  - Derive control laws
  - Test system
  - Generate detailed model
  - Model parts
  - Models suitable for validation
  - Validate by measurements
- **Realization**
  - Gradually enhance laws to code
  - Realization (software)
Embedded Control Software Implementation

• Stepwise refinement
  - Gradually enhance laws to code
  - Gradually add implementation / realization details

• Working order
  - Integrate control laws / sequences
    » Loop control laws + Sequencers + Supervisors
    » Power up / power down sequences
    » Reaction to external commands
    » Implementation assumed ideal
  - Safety, error & maintenance facilities
    » External events on safety (emergency stops)
    » Central or component wise
  - Capture non-ideal components
    » Non-idealness added:
      » Relevant dynamic behavior
      » Signal processing: Estimators
    » Non-idealness Computer hardware
      » HW + SW architecture, Timing aspects
      » Optimization, scheduling

  Mix of
  - Process Structures
  - Control Algorithms

Co-Simulation

• Co-Sim with gCSP / 20sim
  - Formal checks via FDR2
  - Using CTC++ => can be combined with other code
  - Synchronization via separate timer channels / ddl calls

• Co-Sim with POOSSL / 20sim (+TU/e EE)
  - Abstract Syntax, CCS-like
  - SheSim as DT simulator: good timing, but busy waiting – 100% CPU
  - Synchronization between tools via ddl calls

• Co-Sim with VDM++ / 20sim (+RUN / Chess)
  - Distribution / load balancing can be checked
  - Analysis of computer architectures
  - Synchronization between tools via ddl calls

• Explicit simulation of the network
  - Network Simulation via NWsim – rendezvous channels

Working Order
1. Internal checks
2. Formal Check process logic
3. Include (control) algorithms
4. Check target code
Co-simulation gCSP – 20sim I

- Test setup: plotter
  - Testing Embedded Software
- Experiments:
  - Model based design
  - Multiple Views
    » Dynamic model
    » Controller view
    » Mechanical model
    » Fault-tolerant parallel CSP based software
  - Code generation
    » Controller to CPU (RTAI Linux)
    » Controller to FPGA

Co-simulation gCSP – 20sim II

- Software design
  - Verification by co-simulation with 20-sim model

VDM++

- object-oriented formal model-based specification language
- concurrency through threads
- round-trip engineering UML
- formal analysis of static and runtime (type) correctness
- model validation through prototyping & structured testing
- industrial grade tool support
- VICE extension* for real time, scheduling and deployment

* Verhoef, Larsen, Hooman, FM 2006, LNCS 4085, pp 145 - 162
Non-ideal Network: Simulate it

- **CSP approach**
  - Remote Channels couple to Fieldbus
  - Time increment via Timer Channel
  - Packet Simulator based on TrueTime

- **Towards real-time**
  - Remote channel & SimTimer -> Real versions

Network Simulator - Case

- **Simulator OK**
  - Compared with traditional

- **Network parameters**
  - Influence behavior
  - Optimal via simulation

Towards Realization

- **Stepwise refinement**
  - Software-in-the-Loop Simulation
  - Hardware-in-the-Loop Simulation
    - Real-Time Simulation

- **Working order**
  - Part wise towards realization
    - Others still simulated
    - Treatment I/O essential
    - Where, Grouping of functions

- **Useful for**
  - Concurrent engineering
    - Plant simulated
    - SW in time, Plant late
    - Code simulated
    - SW in time, ASIC late
  - Test setup
    - Plant simulated
    - For training purposes
Overview 3rd lecture: Design Method

- Method & ...
  - 4 steps revisited and detailed
  - Focus on Embedded Software -> exercise
- … Techniques
  - Co-simulation
  - Hardware-in-the-Loop Simulation
- Cases
  - On Co-simulation
  - On Hardware-in-the-Loop Simulation
- Overview

Case 1 – HILS setup

- FPGAs as programmable I/O
  - Design I/O functionality as if it were software
  - Easy prototyping
  - Enables Concurrent Engineering
- Tool Chain
  - Effectively download, run, control code

Our Test Set Up

Embedded Control System

- Signal conditioning
- Controller
- Output driver

FPGA

PC104

Sensor simulation
- Model of the plant
- Actuator simulation

Hardware-In-the-Loop simulator

- Electrical interface
- D/A, encoder, etc.
- A/D, PWM, etc.

PC

CAN board

Seco PC/104+ CPU board

RTOS

Embedded Linux, uClibC, RTAI 6 MB

Anything I/O FPGA board

FPGA as programmable I/O

- Design I/O functionality as if it were software
- Easy prototyping
- Enables Concurrent Engineering

Tool Chain

- Effectively download, run, control code
Conclusions Case 1 – HILS setup

- FPGA as I/O is really versatile
  - 1 board: 4 functions: PWM out; Encoder in; PWM in; Encoder out
- HIL-Simulation supports Concurrent Engineering

- Tests
  - Check details
  - Performance

- Refine Design trajectory
  - Optimally benefit from flexibility of FPGAs
- Analogue extension to the FPGA board

Case Study 2 – A/D converter

- Refinement of A/D
  - Essential behavior only
  - Time discretization
  - Functional behavior
    - Quantization (real → integer)
  - Add non-linearities
    - Windowing
  - Nonlinear conversion
  - Add conversion times
    - A/D conversion time considerable

See Paper CCA2001
Case 3 – Production Cell revisited

- **gCSP + 20-sim**
  - Event driven discrete behavior modeled in gCSP + CT library
- **20-sim**
  - 6 Controllers & 12 Motion Profiles modeled & generated (c-code)
  - Event based selection motion profiles
  - Controllers run always
- **Manual coding**
  - Hardware interfacing; c-code in gCSP code blocks

Production Cell in gCSP

- **Top level implementation in gCSP**
  - gCSP stress test

Conclusions – Case 3 – Production Cell

- Prototype tool chain functions rather smoothly
- Shortening design time not (yet) significant
- Continue working on the tools
  - (Try to) combine benefits of all three DT-CT combinations
- Use larger cases in cooperation with Industry

- MovieMinOCW
- MoviePOOSL

NIRICT Kick off – 22-3-07
Overview 3rd lecture: Design Method

- **Method & …**
  - 4 steps revisited and detailed
  - Focus on Embedded Software -> exercise

- **… Techniques**
  - Co-simulation
  - Hardware-in-the-Loop Simulation

- **Cases**
  - On Co-simulation
  - On Hardware-in-the-Loop Simulation

- **Overview**

Topics Design ECS part

- Use the structured design process
- Use integrated models
  - Port-based parts, based on bond graphs and CSP
- Use Simulation as verification means
- Use CSP / gCSP to design Embedded Control Software

- Apply tools and methods on real setup
  - Experiment yourself