EECS 222: Embedded System Modeling Lecture 1

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 1: Overview

- Course Overview
  - Context, content
- Course Administration
  - Schedule, assignments
  - Academic honesty
- Introduction to Embedded System Design
  - Embedded computer systems
  - Levels of abstraction
  - Top-down system design flow
- Project Assignment 1
  - Introduction to application example

Course Context

- Graduate courses on Embedded System Design
  1. **EECS 222: Embedded System Modeling**
  2. **EECS 225: Embedded System Design**
  3. **EECS 226: Embedded System Software**
  4. **EECS 227: Cyber-Physical System Design**

Course Context

- Graduate courses on Embedded System Design
  1. **EECS 222: Embedded System Modeling**

     Computation models for embedded systems.
     System-level specification and description languages.
     Concepts, requirements, examples.
     Embedded system models at different levels of abstraction.
     Modeling of test benches, design under test, IP components.
     Discrete event simulation, semantics, and algorithms.

  2. **EECS 225: Embedded System Design**
  3. **EECS 226: Embedded System Software**
  4. **EECS 227: Cyber-Physical System Design**

Course Context

- Graduate courses on Embedded System Design
  1. **EECS 222: Embedded System Modeling**
  2. **EECS 225: Embedded System Design**

     Embedded system design flow and methodology.
     Design space exploration. Co-design of hardware and software, embedded architecture and network exploration and synthesis.
     System software/hardware interface generation.
     Real-time constraints, specification-to-architecture mapping, design tools and methodologies.

  3. **EECS 226: Embedded System Software**
  4. **EECS 227: Cyber-Physical System Design**

Course Context

- Graduate courses on Embedded System Design
  1. **EECS 222: Embedded System Modeling**
  2. **EECS 225: Embedded System Design**
  3. **EECS 226: Embedded System Software**

     Embedded system software concepts, requirements, examples, for engineering application such as multi-media and automotive.
     Software generation methodology.
     Algorithmic specification, design constraints.
     Embedded operating systems.
     Static, dynamic, real-time scheduling.
     Input/output, interrupt handling.
     Code generation, compilation, instruction set simulation.

  4. **EECS 227: Cyber-Physical System Design**

Course Context

- Graduate courses on Embedded System Design
  1. **EECS 222: Embedded System Modeling**
  2. **EECS 225: Embedded System Design**
  3. **EECS 226: Embedded System Software**
  4. **EECS 227: Cyber-Physical System Design**

     Model-based design of cyber-physical systems including, e.g., plant, sensing, control, actuation, embedded hardware/software, communication, real-time analysis, various levels of simulation (MILS, SILS, HILS), tools and methodologies for automatic synthesis, and application from various interdisciplinary domains.

EECS 222 Course Content

1. Embedded system concepts, abstraction levels, computational models
2. The SpecC system-level description language
3. The SystemC system-level description language
4. Embedded system specification, modeling guidelines
5. Validation, execution and simulation semantics
6. Top-down design methodology
7. System-level architecture modeling
8. Embedded system communication modeling
9. Cycle-accurate modeling, implementation
10. UML and other system-level description languages

Course Administration

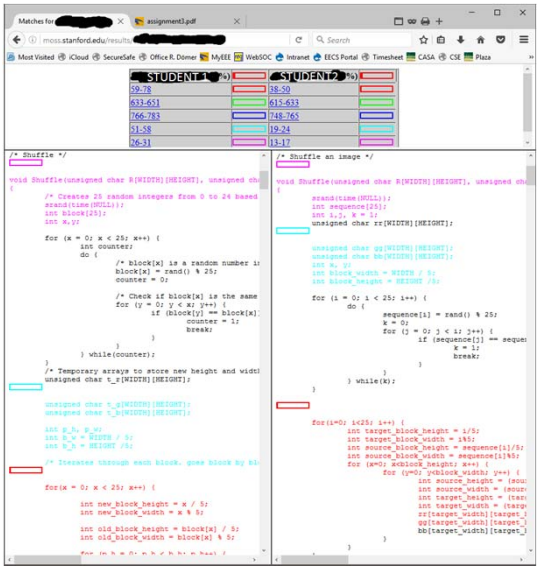
- Course web pages at **`http://newport.eecs.uci.edu/~doemer/w19_eecs222/`**
  - Instructor information
  - Course description and policies
  - Objectives and outcomes
  - Contents and schedule
  - Resources and communication
  - Assignments

Academic Honesty

- Honesty and Integrity are Required
  - See UCI Office of Academic Integrity & Student Conduct
  - See course policy on course web site
- Plagiarism
  - Theft of intellectual property
  - Taking someone else's work or ideas and passing them off as one's own
  - ➢ *Do not copy code!*
- Violations will be reported
  - Academic misconduct report to UCI Office of AISC
    - Interview, written report, AISC staff meeting, decision, …
  - Possible sanctions
    - Warning, probation, suspension, dismissal

Academic Honesty

- Example (F'16):
  - **Moss**: Automatic system for determining similarity of program code



Academic Honesty

- Example (S'17): Text transformation
  - Technical Report 1:

1 INTRODUCTION
With complexities of Systems-on-Chip rising almost daily, the design community has been searching for new methodology that can handle given complexities with increased productivity and less time. The modeling and design of embedded systems can be performed at several abstraction levels. The highest level of abstraction is the System level, where the functionality is described using "system-level specification" (in the case of VLSI design, description languages like VHDL or Verilog) and the architecture is seen as building blocks consisting of processors, memories, etc. …

  - Technical Report 2:

INTRODUCTION:
SOC challenges are changing day by day, the plan group has been hunting down new philosophy that can deal with given complexities with expanded efficiency and less time. The displaying and plan of implanted frameworks can be performed at a few reflection levels. The most abnormal amount of reflection is the System level, where the usefulness is portrayed utilizing "framework level determination" (on account of VLSI plan, depiction dialects like VHDL or Verilog) and the design is viewed as building pieces comprising of processors, recollections, and so on…

Embedded System Design

- Embedded Computer Systems
- System-on-Chip (SoC) Design
- Abstraction Levels
- Embedded System Design Flow

Embedded Computer Systems

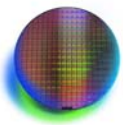
- Computers are ubiquitous, omnipresent…







- *System-on-Chip (SoC) Design*:
  Design of complex embedded systems on a single chip

Embedded Systems

- System embedded into cyber-physical system
  - Constraints from external input (often real-time)
  - Application specific (not general purpose)
- Omnipresent in our environment
  - Pervasive in many application domains
  - In 2005 [Source Netrino]
    - Only 2% of all processors in workstations
    - Remaining 8.8 billion in embedded systems
  - Most computers are embedded systems!


Source: P. Chou, UCI


Source: Edumicator


Source: Miele


Source: Philips

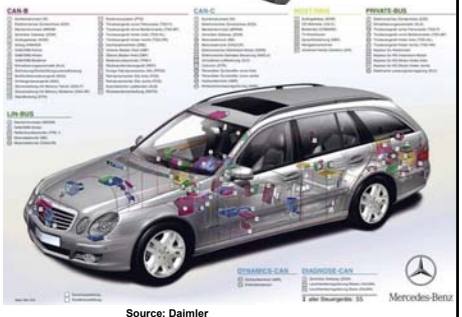

Source: www.trouper.com


Source: www.medicacorp.com/

Embedded System Design

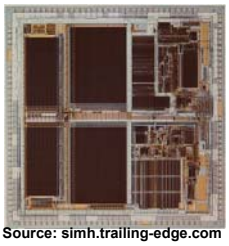
- Design challenges
  - Often mobile
    - Battery powered (low power)
  - Often highly reliable
    - Extreme environment (e.g. temperature)
  - High performance constraints
    - Often real-time requirements
  - High complexity
    - E.g. Mercedes Benz E-class
      - 55 electronic control units
      - 5 communication busses
  - Tightly coupled
    - Software
    - Hardware
  - Rapid development for low price…


Source: Motorola Inc


Source: Daimler

Embedded System Design

- Design Advantages
  - *Application known at design time*
  - *Environment known at design time*
  - Allows for customized / optimized solution
    - Improved performance
    - More functionality
    - At lower power



Source: simh.trailing-edge.com

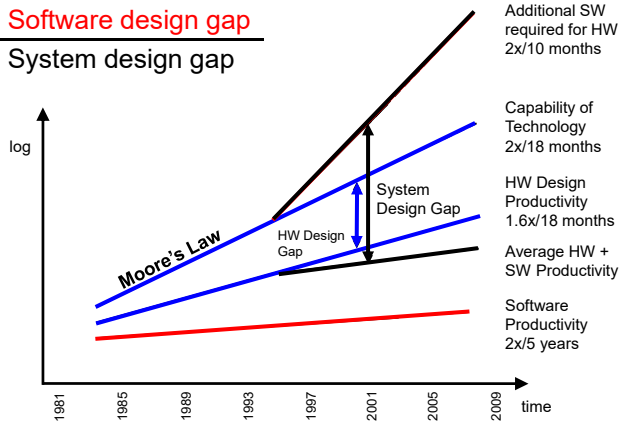
- Custom Platform, SW and HW components
  - Multi-Processor System-on-Chip (MPSoC),
    - Complete embedded system integrated on a chip
  - General-purpose and application-specific processors
  - Application Specific Integrated Circuit (ASIC)
  - Field Programmable Gate Array (FPGA)
  - Circuit board with off-the-shelf-components

Source: Xilinx

Design Complexity Challenge

- Productivity Gap

  Hardware design gap
  + Software design gap
  = System design gap



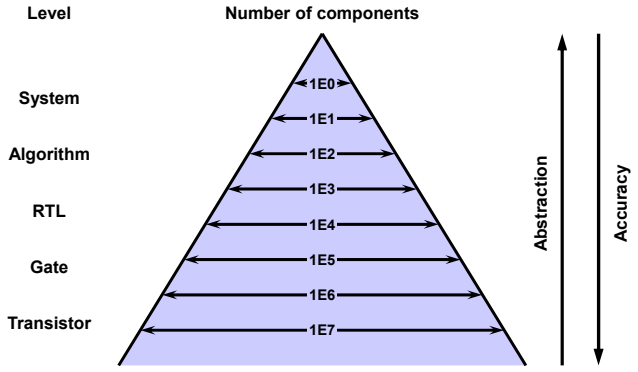
(source: *"Hardware-dependent Software"*, Ecker et al., 2009)

Design Complexity Challenge

- Productivity Gaps
  - Hardware productivity gap
    - Capacities in chip size outpace capabilities in chip design
    - Moore's law: chip capacity doubles every 18 months
    - HW design productivity estimated at 1.6x over 18 months
  - Software productivity gap
    - Growth of SW productivity estimated at 2x every 5 years
    - Needs in embedded SW estimated at 2x over 10 months
  - System productivity gap
    - HW gap + SW gap

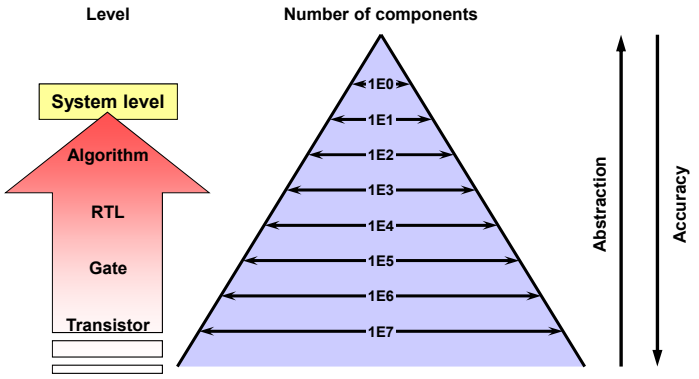
Abstraction Levels

- System-on-Chip (SoC) design faces tremendous increase of design complexity

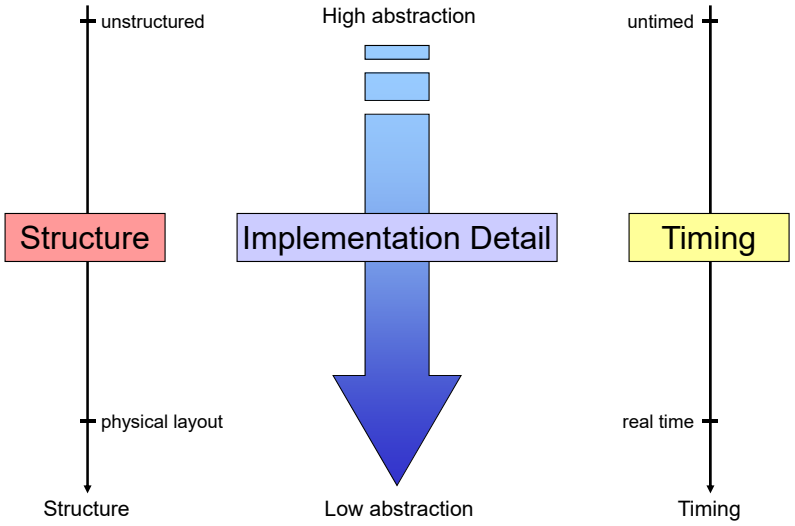


Abstraction Levels

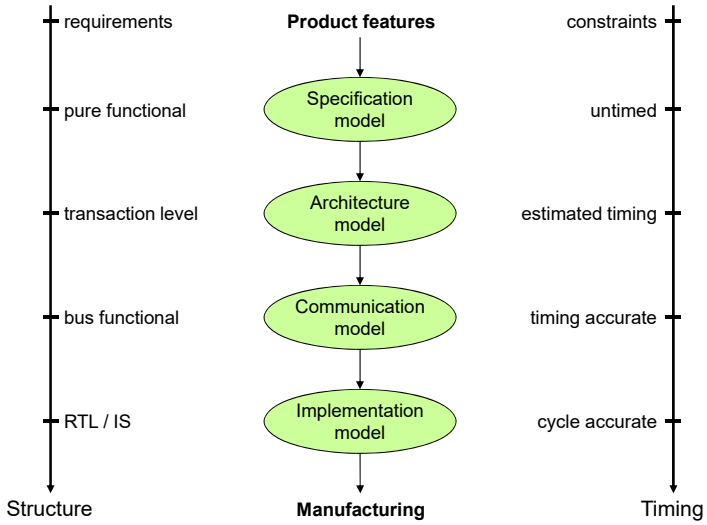
- System-on-Chip (SoC) design faces tremendous increase of design complexity
- Move to higher levels of abstraction!



Abstraction Levels



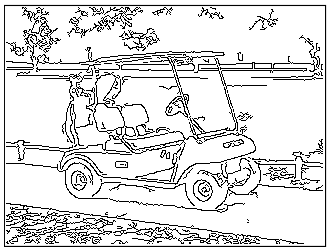
Top-Down System Design Flow



EECS 222 Project

- Application Example: Canny Edge Detector
  - Embedded system model for image processing: Automatic Edge Detection in a Digital Camera

golfcart.pgm



golfcart.pgm_s_0.60_l_0.30_h_0.80.pgm

  - Application Source and Documentation:
    - http://marathon.csee.usf.edu/edge/edge_detection.html
    - http://en.wikipedia.org/wiki/Canny_edge_detector

EECS 222 Project

- Administration
  - EECS Department Linux Servers
    - **`crystalcove.eecs.uci.edu`**, and others
    - Linux environment (CentOS 6.10)
    - Access via secure shell protocol (SSH)
  - Accounts
    - User ID same as your UCInetID
    - Password same as your EEE password
  - Login and make yourself familiar with
    - Command-line tools and GUI tools (which need X client)
    - Text editing and C/C++ programming
    - Image processing tools

Project Assignment 1

- Task: Introduction to Application Example
  - Canny Edge Detector
  - Algorithm for edge detection in digital images
- Steps
  1. Setup your Linux programming environment
  2. Download, adjust, and compile the application C code with the GNU C compiler (**`gcc`**)
  3. Study the application
  4. Fix a bug and clean-up the source code
- Deliverables
  - Source code and text file: **`canny.c`**, **`canny.txt`**
- Due
  - Next week: January 16, 2019, 6pm