EECS 222C: System-on-Chip Software Synthesis 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 administration
  - Communication
- Course overview
  - Context and Contents
  - Objectives and Outcomes
  - Literature
- Introduction to Embedded Systems
  - Overview
  - Characteristics and Applications
  - Embedded Software Issues

Course Administration

- Course web pages at
  `http://eee.uci.edu/10f/18415/`
  - Instructor information
  - Course description
  - Course syllabus
  - Course objectives and outcomes
  - Course resources
  - Assignments
- Course communication
  - Message board
  - Email

Course Context

- EECS 222: Set of 4 courses on SoC Design

A. System-on-Chip Description and Modeling
B. System-on-Chip Design and Exploration
C. System-on-Chip Software Synthesis
D. System-on-Chip Hardware Synthesis

- **Course A is prerequisite for B, C, and D, or consent of instructor**

Course Context

- EECS 222: Set of 4 courses on SoC Design
  - **A. System-on-Chip Description and Modeling**
    Computational models for System-on-Chip (SoC). System-level specification and description languages and execution semantics. Concepts, requirements, examples. SoC modeling at different levels of abstraction (untimed, approximate time, cycle-accurate). Modeling of IP (IP wrappers), design constraints, test benches. Simulation semantics and algorithms. Co-simulation methodology.
  - **B. System-on-Chip Design and Exploration**
  - **C. System-on-Chip Software Synthesis**
  - **D. System-on-Chip Hardware Synthesis**

EECS222C: SoC Software Synthesis, Lecture 1 R. Doemer 5

Course Context

- EECS 222: Set of 4 courses on SoC Design
  - **A. System-on-Chip Description and Modeling**
  - **B. System-on-Chip Design and Exploration**
    System-on-Chip design flow and methodology. Design space exploration. Co-design of hardware and software, hardware/software partitioning. System-on-Chip architecture exploration and synthesis. On-chip network and communication design and synthesis. On-chip software/hardware interface generation.
  - **C. System-on-Chip Software Synthesis**
  - **D. System-on-Chip Hardware Synthesis**

EECS222C: SoC Software Synthesis, Lecture 1 R. Doemer 6

Course Context

- EECS 222: Set of 4 courses on SoC Design

A. System-on-Chip Description and Modeling
B. System-on-Chip Design and Exploration

C. System-on-Chip Software Synthesis
System-on-Chip software concepts, requirements, examples, for engineering applications such as automotive and communication. Software synthesis methodology. Algorithmic specification, design constraints. Applications using embedded operating systems. Static, dynamic scheduling. Input/output, interrupt handling. Code generation, retargetable compilation. Instruction set simulation. Debugging and prototyping.

D. System-on-Chip Hardware Synthesis

Course Context

- EECS 222: Set of 4 courses on SoC Design

A. System-on-Chip Description and Modeling
B. System-on-Chip Design and Exploration
C. System-on-Chip Software Synthesis

D. System-on-Chip Hardware Synthesis
Hardware IP specification. Real-time constraints. Cycle-accurate languages and modeling. Target architectures, data path and control unit. Design tasks and design methodology. Behavioral synthesis. Resource allocation, operation scheduling, binding of operations and variables to functional units, storage units and busses. Communication protocol and interface synthesis. Arbiter, bridge, Transducer, Controller design and synthesis. Net list generation.

Course Contents

- EECS 222C: SoC Software Synthesis
  - System-on-Chip software
    - concepts, requirements, and examples,
    - for engineering applications such as automotive and communication.
  - Software synthesis methodology.
  - Algorithmic specification and design constraints.
  - Applications using embedded operating systems.
  - Static, dynamic, real-time scheduling.
  - Input/output, interrupt handling.
  - Code generation, retargetable compilation.
  - Instruction set simulation.
  - Debugging and prototyping.

EECS222C: SoC Software Synthesis, Lecture 1 R. Doemer 9

Course Goals

- Objectives
  - To learn embedded software concepts in System-on-Chip designs
  - To be able to design, develop and debug software in SoC designs
  - To understand software code generation for SoC
- Outcomes
  - Students understand
    - the special requirements of software for SoC.
    - the process of code generation and integration for SoC.
  - Students are able to
    - develop application SW, middleware, and/or drivers for SoC.
    - implement, test and debug a software application for a SoC.

EECS222C: SoC Software Synthesis, Lecture 1 R. Doemer 10

Course Topics

1 – Embedded software concepts, requirements
2 – SoC software specification, modeling
3 – Embedded software design flow
4 – Real-Time Operating Systems (RTOS)
5 – Real-time requirements, real-time scheduling
6 – Software synthesis, code generation
7 – Hardware-dependent Software (HdS)
8 – Target processors
9 – (Cross-) compilation, execution, debugging
10 – Instruction-set simulation

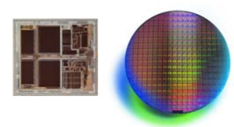
Course Literature

- Primary Textbooks
  - P. Marwedel:
    "Embedded System Design",
    Kluwer Academic Publishers, Boston, 2003.
  - A. Jerraya, S. Yoo, D. Verkest, N. Wehn (editors):
    "Embedded Software for SoC",
    Kluwer Academic Publishers, Boston, 2003.
  - P. Marwedel, G. Goosens (editors):
    "Code Generation for Embedded Processors",
    Kluwer Academic Publishers, 1995.
  - A. Gerstlauer, R. Doemer, J. Peng, D. Gajski:
    "System Design: A Practical Guide with SpecC",
    Kluwer Academic Publishers, Boston, June 2001.
- Additional Reading
  - F. Vahid, T. Givargis:
    "Embedded System Design: A Unified Hardware/Software Introduction",
    John Wiley and Sons, New York, 2002.
  - J. Staunstrup, W. Wolf (editors):
    "Hardware/Software Co-Design: Principles and Practice",
    Kluwer Academic Publishers, Boston, 1997.
  - H. Kopetz:
    "Real-time Systems",
    Kluwer Academic Publishers, Boston, 1997.

Embedded Computer Systems

- Computers are ubiquitous, omnipresent…

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

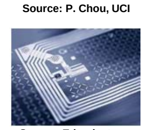


EECS222C: SoC Software Synthesis, Lecture 1 R. Doemer 13

Embedded Systems

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

Source: P. Chou, UCI



Source: Edumicator

Source: Miele

Source: Philips

Source: www.trouper.com

Source: www.medicacorp.com/

EECS222C: SoC Software Synthesis, Lecture 1 R. Doemer 14

Embedded System Design

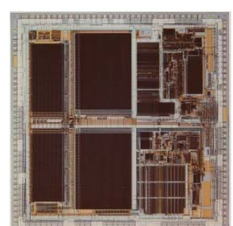
- Design challenges
  - Often mobile
    - Battery powered (low power)
  - Often highly reliable
    - Extreme environment (e.g. temperature)
  - High performance constraints
    - Often real-time requirements
  - High complexity
    - E.g. Mercedes Benz E-class
      - 55 electronic control units
      - 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
- 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



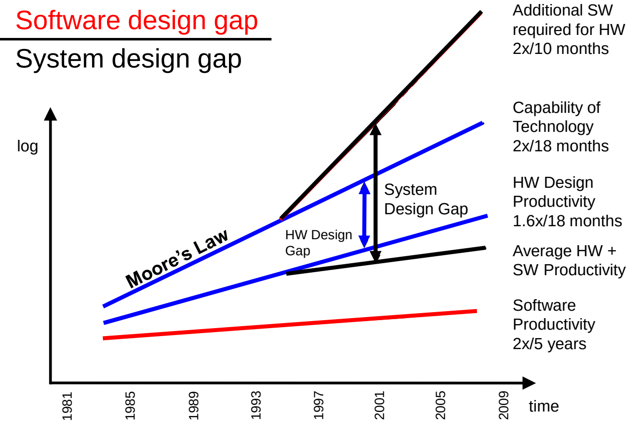
Source: simh.trailing-edge.com

Source: Xilinx

Design Complexity Challenge

- Productivity Gap

  Hardware design gap
  \+ Software design gap
  = System design gap



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

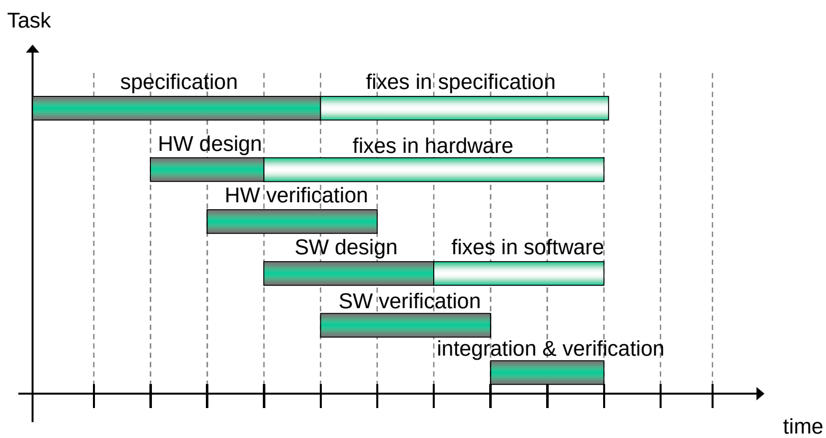
Design Complexity Challenge

- 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

Hardware/Software Codesign

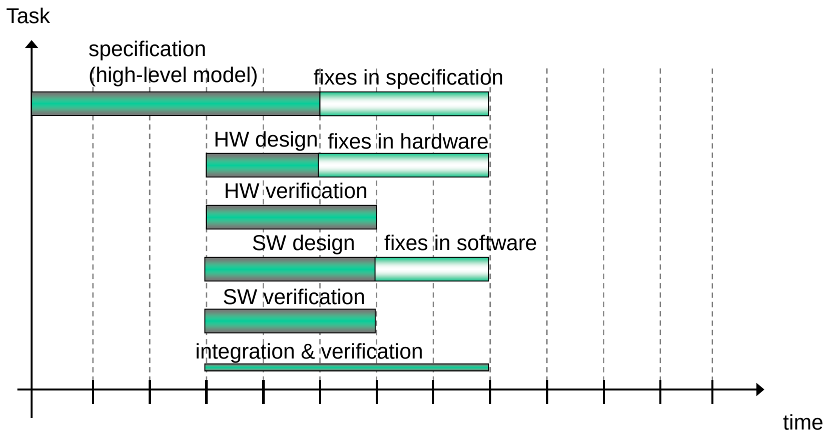
- Traditionally, software development follows hardware
- New: Unified, *concurrent* Design of
  - Hardware and
  - Software
- ➢ Improving Time to Market
  - Faster delivery of new products
  - Higher probability of on time delivery
- ➢ Using a single specification model (System Model)
  - New specification model
  - New specification language
  - ➢ Tight integration of
    - software development
    - hardware development

Traditional Design Flow



Source: Christian Haubelt (U. Erlangen/Nuremberg), Andreas Gerstlauer (U. Texas)

Co-Design Flow (ESL Design)



Source: Christian Haubelt (U. Erlangen/Nuremberg),
Andreas Gerstlauer (U. Texas)

EECS222C: SoC Software Synthesis, Lecture 1 R. Doemer 21

Introduction to Embedded Systems

- Excerpts from Chapter 1 in
  "Embedded System Design"
  by P. Marwedel (Univ. of Dortmund, Germany),
  Kluwer Academic Publishers, 2003.

EECS222C: SoC Software Synthesis, Lecture 1 R. Doemer 22