Embedded Systems Design and Modeling

Introduction

Outline

- Embedded systems:
  - Definition, importance, pervasiveness
- Typical features
- Challenges
- Course goals and objectives
- Course textbook, contents, and grading

Embedded Systems Examples

- Embedded systems have become part of our daily lives ...
  - Entertainment: PDAs, smart cards, players, cameras, set-top boxes, toys, ...
  - Transportation: cars, trains, avionics, mass transit systems, marine, space, ...
  - Communication systems: cell phones, gateways, routers and switches, ...
  - Mission critical systems: nuclear power plant control, elevators, space shuttle, ...

Prime Example: Automotive Electronics

- 400+ computers (*electronic control units, ECUs*) in a premium car today:
  - Engine control, transmission, anti-lock brakes, electronic suspension, parking assistance, climate control, audio system, “body electronics” (seat belt, etc.), display and instrument panel, etc.
  - Nearly 100 Million lines of code
  - Linked together by CAN bus or FlexRay with up to 2km of wiring

What Is An Embedded System?

- No consensus on a clear and complete definition
- But they usually have some typical features and characteristics that distinguishes them from other computing systems and devices
- These features include ...

Embedded Systems Typical Features

- Usually coupled with physical processes: direct interaction with the external world, sensors, actuators
  - Hence sometimes also called “cyber-physical system”
- Hybrid: often contains a mixture of continuous and discrete-state dynamics

Typical Features (Cont'd)

- Reactive:
  1. At the speed of the environment
  2. Real time or at least time-sensitive (passage of time is very important b/c correctness of the results depends on the time at which they are produced)
  3. Inherently concurrent processes competing for resources to perform time-critical tasks

Typical Features (Cont'd)

- Heterogeneous: involves both hardware and software, mixed architectures
  - Application is known a priori
  - But the definition and development of the two occur concurrently (codesign)
  - Some degree of hardware re-programmability might be essential
  - Flexibility in upgrading, debugging, and bug fixing

Typical Features (Cont'd)

- Often networked: remote access, shared
- Often the computational engine of a larger system and not a standalone computer
- Various other constraints exist such as: performance, processing power, power consumption, security, flexibility, reconfigurability, safety, size, weight, heat, reliability, ruggedness, maintainability, etc.

Typical Features (Cont'd)

- Embedded systems employ a combination of
  - application-specific HWs (boards, ASICs, FPGAs, etc.) to reach higher performance and lower power
  - SW on programmable processors (microcontrollers, DSPs, etc.) to gain flexibility while handling complexity
  - mechanical elements (transducers, actuators, sensors)
- Often integrated on a single SoC

Challenges

- HW/SW cosimulation and modeling
- HW/SW partitioning
  - Today: manual approach based on earlier experience with similar products
  - Future: automated, early on, exploring all possibilities
- SW
  - SW design and verification methodology and tools (especially embedded SW) not as developed as HW
  - Satisfying performance constraints with HARD limited resources (registers, memory, time, etc.) ...
  - Need for compositional formalism called “SW synthesis”

SW Challenges (Concurrency)

- Concurrency usually done at the very end and at low level
- Threads and interrupts dominate concurrent software
  - Threads: Sequential computation with shared memory
  - Interrupts: Threads started by the hardware
- Incomprehensible interactions between threads are sources of many problems: deadlock, priority inversion, scheduling anomalies, buffer overruns, non-determinism, system crashes

SW Challenges (Being Real-Time)

- All our computation and programming abstractions are built on this premise:
  - Correct execution of a program in programming languages (C, C++, C#, Java, etc.) has nothing to do with how long it takes to do the tasks
  - Timing of programs is not precisely repeatable
- Conclusion: programmers have to step outside the programming abstractions to specify timing behavior

SW Challenges (Verification)

1999: NASA lost a $125 million Mars orbiter because:

- one engineering team used metric units while another used English units for a key spacecraft operation!
- due to an un-initialized variable!

SW Mishaps (Cont'd)

- Recently: SW bugs cause sudden engine stalls in high-tech cars!
- U.S. Department of Commerce National Institute of Standards and Technology (NIST): Software bugs cost the U.S. economy an estimated $59.5 billion per year!
- No methodical SW verification approach yet!

Challenges (Power)

- Bottom-up view: battery life time, cooling chips as hot as the surface of Sun
- Top-down view: energy is the upper bound on the available computations
  - Total Milky Way energy: 10^{59} J
  - Minimum switching energy: 1.6×10^{-20} J
  - Max. number of operations: 6×10^{78}
  - Current number of operations/year: 3×10^{24}
  - Assuming doubling of computations each year:
  - All energy will be consumed in 180 years!

Challenges (HW)

- HW design
  - Many implementation choices: microprocessors, microcontrollers, DSPs, network processors, ASIPs, reconfigurable architectures, ASICs, FPGA, etc.
  - Managing complexity of parallel architectures
  - Automated verification tools needed at all levels
  - Reliability
  - System integration
  - Productivity gap

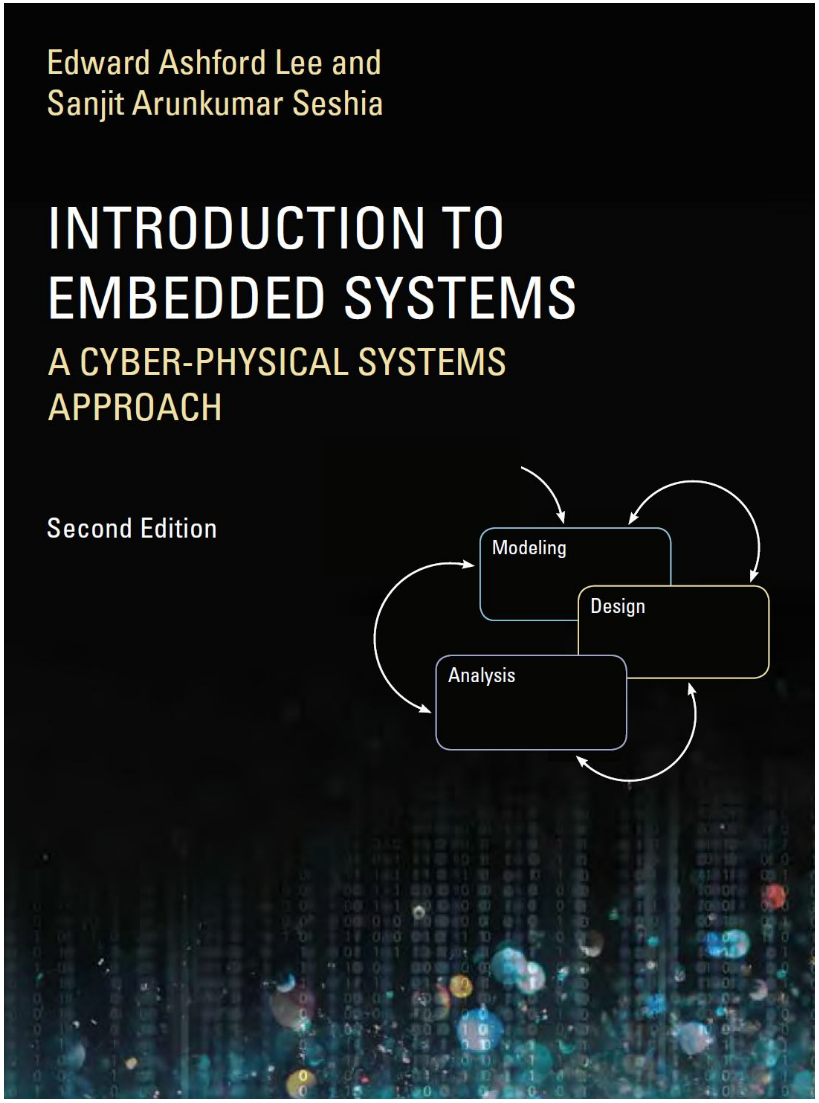
Conclusions

- Embedded systems: fact of life
- 2D challenges matrix ahead of us:
  - Complexity, heterogeneity, design, verification
  - SW, HW, Power
- We need a new formal foundation for embedded systems to:
  - precisely model these systems
  - systematically and even-handedly recombine computation and physicality
- This course attempts to address these needs

Course Goals and Objectives

- A principled, scientific approach to modeling and design of embedded systems
- Not just trial and error, which can be very painful when things go wrong
- There will be a focus on:
  1. Modeling and model-based system design
  2. General principles of embedded software design (mainly scheduling and concurrency)
  3. Analysis of embedded systems at higher levels of abstraction

Textbook

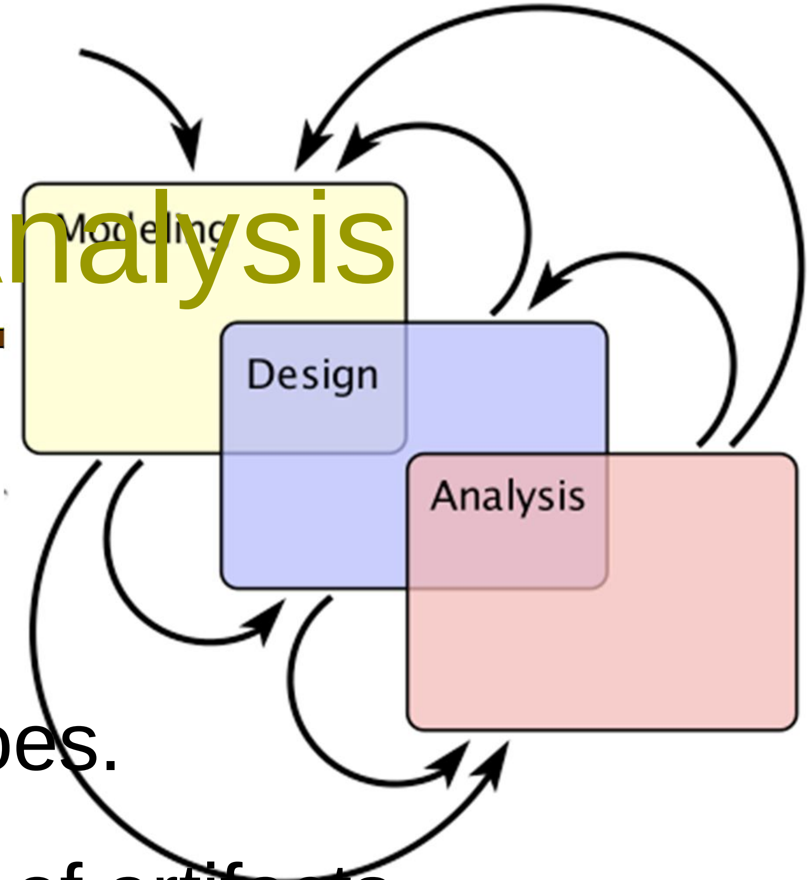


- E. A. Lee and S. A. Seshia, Introduction to Embedded Systems, A Cyber-Physical Systems Approach, 2nd Ed., 2017
- The textbook introduces the intellectual ideas of embedded systems as a technology and as a subject of study
- http://LeeSeshia.org
  - Soft copy available at the website or through instructor

Textbook Parts

- The book has three main parts:
  - Part I (chapters 2-6) focuses on modeling the cyberphysical systems and introduces various models of computation (MoC)
  - Part II (chapters 7-12) deals with design aspects and hardware elements of embedded systems
  - Part III (chapters 13-17) concentrates on analysis of embedded systems
- We will cover parts I and III completely
- Part II will be covered partially

Modeling, Design, Analysis



Modeling is the process of gaining a deeper understanding of a system through imitation. Models specify **what** a system does.

Design is the structured creation of artifacts. It specifies **how** a system does what it does.

Analysis is the process of gaining a deeper understanding of a system through dissection. It specifies **why** a system does what it does (or fails to do what a model says it should do).

Motivating Example of a Cyber-Physical System

Modeling:
- Flight dynamics (ch2)
- Modes of operation (ch3)
- Transitions between modes (ch4)
- Composition of behaviors (ch5)
- Multi-vehicle interaction (ch6)

Design:
- Processors (ch7)
- Memory system (ch8)
- Sensor interfacing (ch9)
- Concurrent software (ch10)
- Real-time scheduling (ch11)

Analysis
- Specifying safe behavior (ch12)
- Achieving safe behavior (ch13)
- Verifying safe behavior (ch14)
- Guaranteeing timeliness (ch15)

Course Contents

- The course consists of:
  - Lectures based on the textbook and relevant papers that cover the conceptual basis
  - Homework assignments from the book (25%)
  - Research report based on the most recent research papers (25%)
  - Final exam (25%)
  - Final project (25%)