Fundamental Algorithms for System Modeling, Analysis, and Optimization

Jaijeet Roychowdhury, Stavros Tripakis
UC Berkeley
EECS 144/244
Fall 2014

Copyright © 2010-date, E. A. Lee, J. Roychowdhury, S. A. Seshia, S. Tripakis, All rights reserved

Lecture 1: Introduction, Logistics

Models and Algorithms in the Design & Analysis of Complex Systems
Complex systems

$10^{11}$ stars

$10^{100,000}$ states
A simple program

```plaintext
int x := input an integer number > 1;

while x > 1 {
    if x is even
        x := x / 2;
    else
        x := 3*x + 1;
}
```
A simple program?

```c
int x := input an integer number > 1;

while x > 1 {
    if x is even
        x := x / 2;
    else
        x := 3*x + 1;
}
```

Collatz conjecture: the program terminates for every input. Open problem in mathematics.

Safety-critical systems

- Smart cars, roads, buildings, power grid, cities, ...
How to design such systems?

Designers need tools!

=> computer-aided design

Computer-Aided Design (CAD) for ICs / Electronic Design Automation (EDA)

731M transistors
Approaches to system design (1)

- **Trial-and-error** approach:
  - Build prototype
  - Test it, find errors
  - Fix errors
  - Repeat

Design by trial-and-error

- Toyota unintended acceleration incidents
- Millions of cars recalled
- Cost: $ billions
- U.S. National Highway Transportation Safety Administration’s (NHTSA) report concluded that electronic throttle control systems were not the cause.
Design by trial-and-error

- Boeing 787 grounded
- “All-Nippon today announced it had canceled 320 flights, including 51 international flights, on 787s affecting a total of 46,800 passengers” [San Jose Mercury News, 1/22/2013]
- FAA restriction finally lifted in April 2013.

As a result of an in-flight, Boeing 787 battery incident earlier today in Japan, the FAA will issue an emergency airworthiness directive (AD) to address a potential battery fire risk in the 787 and require operators to temporarily cease operations. Before further flight, operators of U.S.-registered Boeing 787 aircraft must demonstrate to the Federal Aviation Administration (FAA) that the batteries are safe.

Design by trial-and-error

- Last but not least ...
Design by trial-and-error

- Software!

Approaches to system design (2)

- Rigorous, “model-based” design:
  - Build model (“executable specification”) of system
    - Before building a prototype of the system itself
  - Analyze the model, find errors
  - Fix errors in the design (model)
  - Repeat until the design seems OK
  - Give models/specs to someone (or to a computer) to implement them
    - Need to ensure properties are preserved during implementation

- Better for affordability:
  - Catch design errors early => easier / less costly to fix

- Better for dependability:
  - Sometimes can formally prove that design is correct

- Gaining acceptance in the industry
The Elements of Model-Based Design

How to describe what we want?

How to be sure that this is what we want?

Modeling

Analysis

Implementation, Optimization

How to build it? Automatically Correct-by-construction

From standard compilers ...

class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}

source code
(C, Java, ...)

compiler

type checking, debugging, static analysis, ...

machine code
… to system compilers

Vision: modeling/simulation languages of today will become the system-programming languages of tomorrow

Rich languages: concurrency, time, robustness, reliability, energy, security, …

Powerful analyses: model-checking, WCET analysis, schedulability, performance analysis, reliability analysis, …

Complex execution platforms: networked, distributed, multicore, …

Caveat

In real life, we need both MBD and trial-and-error methods. Why?

1. We cannot trust our models 100%
2. All models are abstractions of reality. They make assumptions that need not hold.
   - E.g., road condition, weather condition, …
3. Analysis and optimization methods also have their limitations.
   - As we will see in this course.
Model-based design seems fine, but …

- There are many systems, of different kinds
- People have been designing these for decades
- Can we pretend to find a single design method that works for every kind of system?
- Of course not
- Thesis:
  - System design is a science
  - There is a body of knowledge (models, algorithms, …) which is fundamental to that science
  - This body of knowledge is applicable to many application domains (circuits, SW, embedded systems, bio, …)

Example of a successful model-based design flow

RTL synthesis flow

K. Keutzer
CAD at Berkeley: History

- CAD research at Berkeley: design tools with an impact
  
  late 60s and 70s
  CANCER, SPICE (Rohrer, Nagel, Cohen, Pederson, ASV, Newton, etc.)
  SPLICE (Newton)

  80s
  MAGIC (Ousterhout et. al.)
  Espresso (Brayton, ASV, Rudell, Wang et. al.)
  MIS (Brayton, ASV, et. al.)

  90s
  SIS, HSIS (Brayton, ASV et. al.)
  VIS (Brayton, ASV, Somenzi et. al.)
  Ptolemy (Lee et. al.)

  2000-date
  MVSIS (Brayton, Mishchenko et. al.)
  BALM (Mishchenko, Brayton et. al.)
  ABC (Mishchenko, Brayton et. al.)
  MetroPolis, Metro II, Clotho (ASV et. al.)
  Ptolemy II, HyVisual (Lee et. al.)
  UCLID, GameTime, Beaver (Seshia et. al.)

Lecture Outline

- Introduction to Jaijeet, Stavros, and all of you
- Some of the topics covered in this course
  - Digital systems (circuits)
  - Cyber-Physical systems
  - Continuous-time systems
- Course logistics
Jaijeet Roychowdhury

- **Past:**
  - PhD: UCB EECS (1993)
  - Bell Labs (1993-2000)
  - CeLight (2000-2001)
  - Univ. of Minnesota (2001-2008)

- **Research interests:**
  - modelling/simulation of continuous-time systems
  - widely-separated time scale problems
  - noise and variability
  - automated macromodelling

Stavros Tripakis

Adjunct Assoc. Prof. and Researcher
UC Berkeley (2009 – now)

- **Past:**
  - Postdoc: Berkeley, 1999 – 2001
  - PhD: Verimag Laboratory, Grenoble, France, 1998
  - Undergrad: University of Crete, Greece, 1992

- **Research interests**
  - System design, modeling, and analysis (DMA)
  - Formal methods
  - Computer-aided verification and synthesis
  - Compositionality, contracts, interfaces
  - Embedded and cyber-physical systems
Round of introductions

Your name, research/professional interests, grad/undergrad student, ...

Course topics

Algorithms for Discrete Models

• Automata, state machines, transition systems, logic, temporal logic
• State-space exploration, reachability analysis, model-checking
• Boolean function representation and manipulation
• Synchronous and asynchronous composition

Algorithms for Continuous Models

• Solving non-linear equations

Algorithms for Cyber-Physical Models

• Timed and discrete-event systems
• Discrete-event simulation

Cross-cutting Topics

• Timing analysis
• Controller and program synthesis
Part I: Digital Systems

Evolution of Digital IC Design

Results
(Design Productivity)

Effort
(EDA tools effort)

Transistor entry

Schematic Entry

RTL Synthesis

1978

1985

1992

1999

What’s next?

McKinsey S-Curve

K. Keutzer
Evolution of the EDA Industry

Results (Design Productivity)

1978
1985
1992
1999

Synthesis - Cadence, Synopsys
Schematic Entry - Daisy, Mentor, Valid
Transistor entry - Calma, Computervision

McKinsey S-Curve

Effort (EDA tools effort)

What’s next?

Transistor Era

Key tools:

- Transistor-level layout – e.g. Calma workstation
- Transistor-level simulation – e.g. Spice
- Bonus: transistor-level compaction – e.g. Cabbage

Size of circuits: 10’s of transistors to few thousand

Key abstractions and technologies:

- Transistor-level modeling, simulation
- Logical gates- NAND, NOR, FF and cell libraries
- Layout compaction

K. Keutzer
Gate-level Schematic Era

Key tools:

- gate-level layout editor – Daisy, Mentor, valid workstation
- Gate-level simulator
- Automated place and route

Size of circuits: 3,000 – 35,000 gates (12,000 to 140,000 transistors)

Key abstractions and technologies:

- Logic-level simulation
- Cell-based place and route
- Static-timing analysis

RTL Synthesis Era

Key tools:

- Hardware-description language simulator – Verilog, VHDL
- Logic synthesis tool - Synopsys
- Automated place and route – Cadence, Avant!, Magma

Size of circuits: 35,000 gates to …?

Key abstractions and technologies:

- HDL simulation
- Logic synthesis
- Cell-based place and route
- Static-timing analysis
- Automatic-test pattern generation
- Equivalence checking / verification
RTL Synthesis Flow

K. Keutzer

Part II: Cyber-Physical Systems
Embedded, Cyber-Physical Systems

- Computers (HW+SW) “embedded” in a physical world
- Cyber = computers (literally “to govern”)
- Physical = the rest
- Typically in a closed-loop (feedback) control configuration (often there are many distributed controllers)
CPS Example – Printing Press

- High-speed, high precision
  - Speed: 1 inch/ms
  - Precision: 0.01 inch
    -> Time accuracy: 10us
- Open standards (Ethernet)
  - Synchronous, Time-Triggered
  - IEEE 1588 time-sync protocol
- Application aspects
  - local (control)
  - distributed (coordination)
  - global (modes)

Another Example of a CPS Application

Modeling:
- Flight dynamics
- Modes of operation
- Transitions between modes
- Composition of behaviors
- Multi-vehicle interaction

Design:
- Processors
- Memory system
- Sensor interfacing
- Concurrent software
- Real-time scheduling

Analysis
- Specifying safe behavior
- Achieving safe behavior
- Verifying safe behavior
- Guaranteeing timeliness

Bosch-Rexroth

Edward A. Lee

STARMAC quadrotor aircraft (Tomlin, et al.)
CPS challenges

How to model CPS? What are suitable mathematical formalisms which combine discrete & continuous dynamics?

How to analyze (simulate, verify, …) such models?

How to derive (ideally, automatically) valid implementations from the models?

Part III: Continuous-Time Systems

See separate set of slides by Jaijeet Roychowdhury
Course Logistics

Webpage, Books, etc.
The course webpage is the definitive source of information

http://embedded.eecs.berkeley.edu/eecsx44/

We’ll also use **bCourses** (not bSpace)

No textbook. Readings will be posted / handed out for each set of lectures.

Some references will be placed on reserve in Engineering library.

GSI: Chris Shaver

See webpage for office hours, etc.
Format of Lectures

2 1.5 hour lectures per week (Mon-Wed 2:30 – 4 pm)
1 hr Discussion section / lecture each Mon 4-5 pm
  • usually a topic supporting homeworks/projects; sometimes an extension of lectures

Grading

2 Midterms (20% each)
8-10 homework assignments (total ~ 25%)
1 course project (~ 35%)

Course project:

- Graduate students (244): Must investigate a novel research idea
- Undergraduates (144): Encouraged to do 244-style project (join with grads!); also permitted to do implementation projects or literature surveys