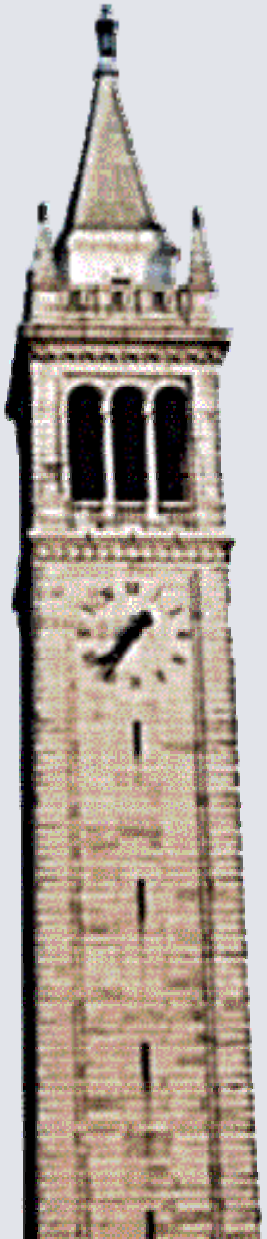
# A New Course in Hybrid and Embedded Systems

Edited and presented by
Claire J. Tomlin
UC Berkeley



Chess Review
November 21, 2005
Berkeley, CA

# Background

Currently:

- **[Berkeley]** EECS 291e/ME 291S

  Hybrid Systems: Computation and Control

  **http://robotics.eecs.berkeley.edu/~sastry/ee291e/HSCC05.htm**

- **[Stanford]** AA 278A

  Hybrid Systems: Modeling, Analysis, and Control

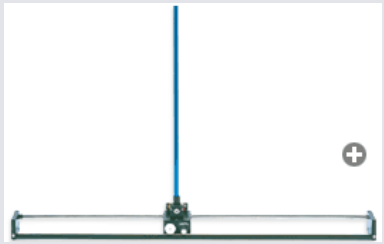
  **http://www.stanford.edu/class/aa278a/**

# Current coverage includes:

- Modeling: finite state machines, ODEs, PDEs
- Bisimulations: timed automata
- Stability analysis for switched systems
- Reachability analysis for verification and controller synthesis
- Tools:
  - HyTech
  - PtolemyII
  - HyVisual
  - Level Set Toolbox
  - Requiem, d/dt, CheckMate

# Projects

- Course projects
  - In depth analysis or design in any of the areas covered in class
  - Midterm: preliminary design review
  - Final: a short (10 page) paper and 15-20 minute presentation of project and results
- Examples
  - Hybrid optimization applied to path planning
  - Stability analysis and control of inverted pendulum
  - Lane-keeping/lane-changing control
  - Autonomous space station rendezvous and docking
  - Multimode engines (HCCI)
  - Four person soccer game
  - Pursuit evasion with 2 (time varying) players



# Proposed New Course

- "Mezzanine" level course in hybrid and embedded systems
- Open to upper level undergraduates, and graduate students
- Lectures, and strong focus on **semester-long project**
- Project based on real examples developed with Industry partners
  - Examples: Boeing DemoSim, "Airbus-like" code
- Full fledged course in 2006-07 (though some components tested in Spring 2006)

# Needs: a stronger link with design cycle for real embedded systems

- A methodology that integrates verification, validation, and test procedures throughout the requirements, design, implementation, and testing cycle:
  - High level design tools (such as HyVisual, Ptolemy II, Simulink) for specifying the semantics of these components and their interfaces, generation of corresponding test suites
  - High level programming language (like C, Java) to implement these components, generation of corresponding test suites
  - Automatic code generation tools (ideally, for both the high level code and the low level implementation)
  - Automatic test suite generation
- How to bring certification into the classroom?

  [Frank McCormick, Certification Services, Inc.]

# Timeline

- Full fledged course in 2006-07 (some components tested in Spring 2006)
- Request input and mentorship from industrial partners for individual project design
  - Automotive
  - Aviation
  - Industrial automation
  - Manufacturing
  - Critical infrastructures (California Energy Commission)