Final Project

- Schedule, Organization
- Choosing a topic
- Example projects
- Grading
- Design Suggestions

Final Project: Schedule

- Choose project teams (email cjt by Mon, Oct.20)
  - Teams of two (or maybe three). A single person project requires approval of lecturer.
- Project Abstract (due Mon, Oct. 27, submit on-line)
  - Start discussing ideas now with 6.111 staff
  - About 1 page long, clearly identify who's doing what
- Proposal Conference with staff mentor (by Fri, Oct. 31)
  - Bring your proposal with you *and* submit on-line
- Block Diagram Conference with mentor (by Fri, Nov. 7)
  - Review major components and overall design approach
  - Specify the device components you need to acquire (*small* budget allocated for each project if component does not exist in the stock room). Get approval from the 6.111 staff and your TA will contact John Sweeney to obtain the parts.

Schedule (cont'd.)

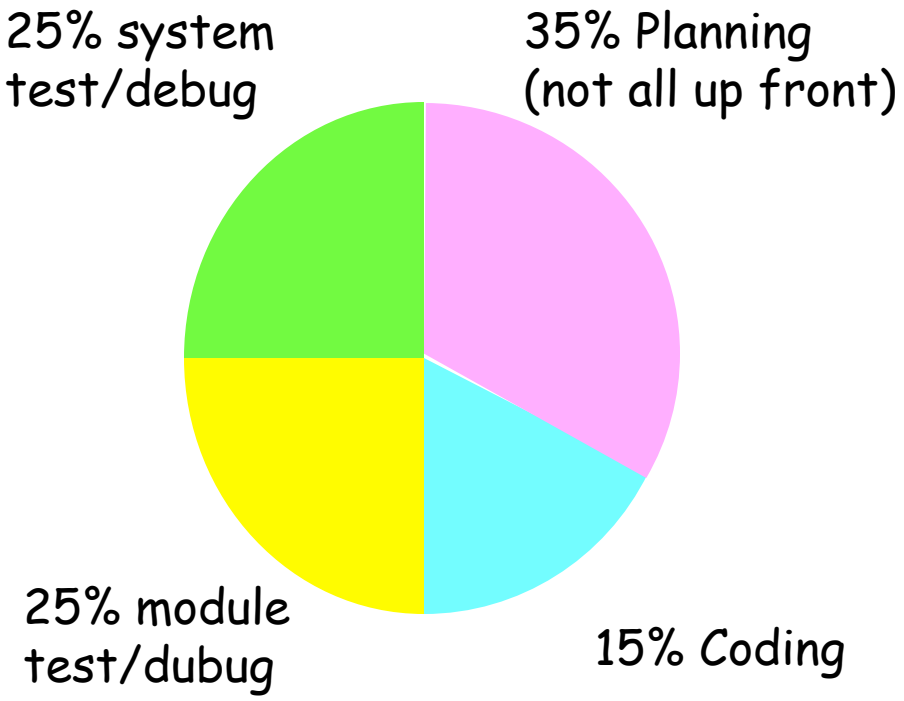
- Project Design Presentation to staff (Nov 12&13)
  - Each group will make a 15 min electronic presentation (~10 slides) dividing presentation among team members
  - Submit PDF on-line, will be posted on website
  - Example: S2004 FROGGER presentation slides
- Project Checkoff Checklist to staff (Nov 14)
  - Each group in discussion with TA creates a checklist of deliverables (i.e., what we can expect each team member to demonstrate). Submit PDF on-line.
- Final Project Presentations (Dec 8-10)
  - Videotaped and posted on-line with your permission
- Final Project Report (Wed, Dec 10, 5p)
  - Submit PDF on-line, will be posted on website
  - Sorry, no late checkoffs or reports will be accepted

Team Organization

- Most importantly, you need one
- Key decisions made jointly
  - Requirements
  - High level design
  - Schedule
  - Who will work on what, who'll take the lead
  - Response to slippage
- Lower level design exchanged for examination
  - Everyone responsible for everything
  - Design reviews tremendously helpful
    - Try it, you'll like it
- Communicate with each other early and often

Controlling Schedule

- First, you must have one
- Need verifiable milestones
- Some non-verifiable milestones
  - 90% of coding done, 90% of debugging done, Design complete
- Need 100% events
  - Module 100% coded, Unit testing complete
- Need critical path chart
  - Know effects of slippage
  - Know what to work on when



Provide a 4-7 day contingency to deal with unforeseen issues (you'll use it all!)

Choosing A Topic

- You only have 6 weeks total (once your proposal abstract is turned in) to do this project.
  - It is important to complete your project.
  - It is very difficult to receive an "A" in the class without having something working for the final project.
- The complexity for each team member should greater than the complexity of the lab assignments.
- Some projects include analog building blocks or mechanical assemblies (infrared, wireless, motors, etc.). However, keep in mind that this is a digital design class and your design will be evaluated on its digital design aspects.
- Complexity, risk and innovation factor.
  - We will give credit to innovative applications, design approaches
  - More complex is not necessarily better
- Look through previous projects for inspiration (see website)

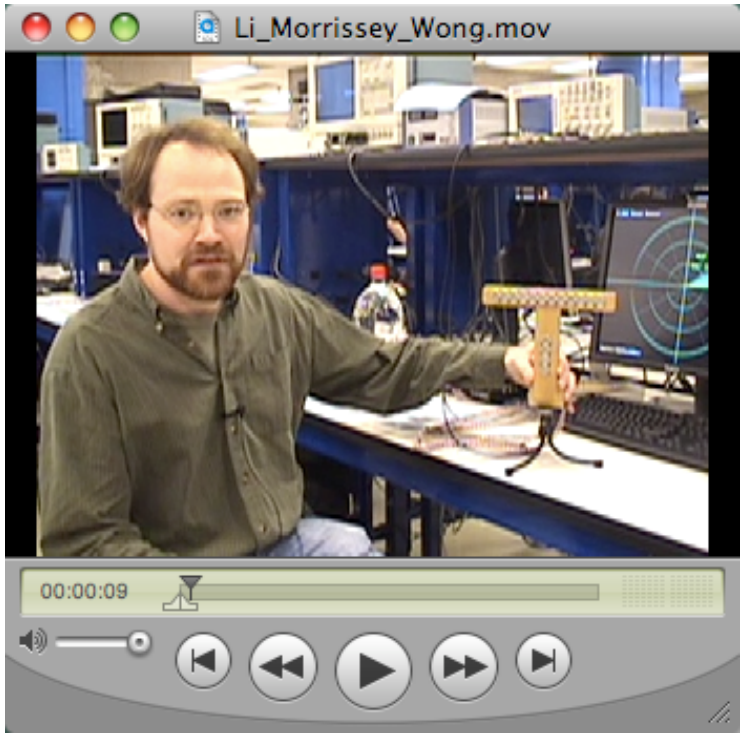
Example Projects

Music Transcriber
Roberto Carli, Alessandro Yamhure
Fall 2005

Virtual Juggling
David Rush, Christopher Wilkens
Fall 2005

Example Projects

Conductor Hero
Natalie Cheung
Yuta Kuboyama
Edgar Twigg
Fall 2007



A Simple Digital Sonar System
Zhen Li
Bryan Morrissey
Brian Wong
Fall 2007

Example Projects

Raytracer
Zhen Li
Sam Gross, Adam Lerer
Spring 2007

Some Suggestions

- Be ambitious!
  - But choose a sequence of milestones that are increasingly ambitious (that way at least part of your project will work and you can debug features incrementally).
  - But don't expect 400Mhz operating frequencies, etc.
- It's motivating if there's something to see or hear
  - Video and graphics projects are fun (and with the labkit basic video input and output are pretty straightforward which means you can concentrate on the processing)
  - Audio/Music is low-bandwidth, so it's easy to do interesting processing in real-time (real-time is harder with video).
- Memories are often the limiting factor
  - Figure out how you'll use memory blocks early-on

More Suggestions

- Be modular!
  - Figure out how test your modules incrementally (good for debugging and checkoff!)
  - Be clear about what information is passed between modules (format, timing)

- Don't be caught by the mañana principle
  - Six weeks goes by quickly: have a weekly task list.
  - How does a project run late: one day at a time!
  - Effort is not the same as progress: "Written but not tested" only means you've made a start
  - Tasks will take longer than you think
  - Final integration will uncover bugs/thinkos so test module-to-module interactions as early as you can

Grading (40 points Total)

- Report and Presentation (5 points)
- Deadlines and Participation (5 points)
- Problem Definition and Relevance, Architecture, Design methodology (10 points)
  - What is the problem
  - Why is it important
  - System architecture and partitioning
  - Design choices and principles used
  - Style of coding
  - All of the above should be stated in the project and report
- Functionality (10 points)
  - Did you complete what you promised (i.e., graded by the checklist)
- Complexity, Innovation, Risk (10 points)

Design Suggestions

- Use hierarchical design
  - Partition your design into small subsystems that are easier to design and test.
  - Design each sub-system so they can be tested individually.
  - When appropriate, use Major/Minor FSMs.
- Use the same clock edge for all edge-triggered flip-flops
  - Beware of clock skew, don't gate the clock
  - If you have multiple clock domains, think very carefully about how you transfer information from one to another
- Avoid problems from 'glitches'.
  - Always assume that combinational logic glitches
  - Never drive a critical asynchronous control signal (register clock, write enable) from the output of combinational logic.
  - Ensure a stable combinational output before it is sampled by CLK.
  - When needed, create glitch-free signals by registering outputs.

Design Suggestions (cont'd.)

- Avoid tri-state bus contention by design
- Synchronize all asynchronous signals
  - Use two back-to-back registers
- Use asynchronous memories properly
  - Avoid high Z address to SRAM when CE is asserted.
  - Avoid address changes when WE is true.
  - Make sure your write pulse is glitch free.
- Use care when incorporating external devices
  - Use bypass capacitors on external components to deal with noise
  - I/O pads are slow, not all signals have the same delay
- Chip-to-chip communication
  - Beware of noise (inductance)
  - Might need to synchronize signals
  - Can also use "asynchronous" protocols