ECE 485/585
Microprocessor System Design

Lecture 1: Course Overview
Finite State Machine Design in Verilog

Zeshan Chishti
Electrical and Computer Engineering Department
Maseeh College of Engineering and Computer Science
PSU Westside Program

- **Goal:** Provide a full MS ECE program targeted to full-time professionals (and students) working/living in the Silicon Forest:
  - Courses are offered during the evenings
  - Courses are offered at the Willow Creek Center in Hillsboro
  - Courses focus on gaining practical hands-on experience in addition to theoretical knowledge
  - Courses are taught by instructors with industry expertise in the areas we’ll be studying
  - Students can earn an MS degree only taking courses on the Westside

- **WCC Graduate track focus is on:**
  - IC design, test and validation
  - Embedded systems
  - Computer architecture and design
WCC Lab

- Located in WCC 313
- Four lab stations. Each station is equipped with:
  - PC with commonly used design tools (Vivado, Android, QuestaSim, ...)
  - Tektronix MDO3000 Series Oscilloscope/Spectrum Analyzer/Logic Analyzer/Waveform generator
  - Tektronix DMM 4020 5 ½ digit multimeter
  - Keithley triple DC power supply
- Collaboration space for team projects
- Wifi (PCC) and VPN Tunnel to PSU
  - Log in to the station w/ your MCECS Id/Password
Course Information

- **Instructor**
  - Zeshan Chishti, zeshan@pdx.edu, 503.720.6390 (C)

- **TA**
  - Aditi Gundu, adigundu@pdx.edu

- **Grader**
  - Rushabh Chavan, rushabh@pdx.edu

- **Office Hours:**
  - Zeshan:
    - Before/after class or by appointment
  - Aditi: TBD

- **Class meets Monday and Wednesday from 5:00PM – 6:50 PM in WCC 310**
Prerequisites

- ECE 372 – Microprocessor Interfacing and Embedded Systems
- Programming language such as C or C++ is helpful but not required
- Working knowledge of Verilog or VHDL is helpful but not required
Course Website

- http://ece.pdx.edu/~zeshan/ece585.htm
- Single source for all the course-related information
  - Announcements, latest syllabus and course calendar
  - Lecture notes
  - Homework assignments
  - Links to video recordings
  - Links to resources
Course Objectives

- Understand and be able to design microcomputer-based systems
- Understand and be able to design static, dynamic, and non-volatile memory systems
- Understand and be able to model multilevel cache and virtual memory systems
- Understand the architecture, advantages and disadvantages of common PC buses such as PCI and USB
- Gain experience working as a member of a project team implementing a complex modelling project
- Gain experience making good technical presentations
The Big(ger) Picture

☐ This course is:
  ■ Part of a four course sequence
    ☐ ECE 485/585 – Microcomputer System Design
    ☐ ECE 486/586 – Computer Architecture
    ☐ ECE 587/687 – Advanced Computer Architecture I
    ☐ ECE 588/688 – Advanced Computer Architecture II
  ■ Included in several of the Graduate Tracks*
    ☐ Core course for Computer Architecture and Embedded System tracks
    ☐ Depth/Breadth course for Digital IC Design, Test and Validation

☐ This course is not:
  ■ A course in microprocessors (that’s ECE 371)
  ■ A course in interfacing to microprocessors (that’s ECE 372)
  ■ A course in Verilog (that’s ECE 351)
  ■ A course in FPGA design (that’s ECE 540 and ECE 544)
  ■ A course in Embedded System Programming (that’s ECE 558)

* http://pdx.edu/ece/graduate-tracks
Quick Survey

☐ Taken an operating systems course?
☐ Do you know what an FPGA is?
☐ Which microprocessor families are you familiar with?
   - Intel x86
   - ARM
   - MIPS
   - Other
☐ C/C++ Programming?
☐ Verilog? VHDL? System Verilog?
☐ Have you heard of, can you define/explain...
   - Two way set associative cache
   - Demand paged virtual memory
   - Vectored interrupt
   - TLB
ECE 485/585 Course Organization

- Lectures:
  - Topics listed in syllabus – you are expected to have completed required readings (if any) ahead of time
  - Lecture notes will be posted to course website at least a few hours before class

- Homework and final project:
  - Homework assignments will be posted to the course website; assign/due dates in the course schedule on the course website
    - All homework assignments will be due by 5PM on the due date
    - Students can submit via hardcopy (in class) or electronically
    - We’ll try to return feedback/grades within 2 weeks after due date
  - Final project assigned week 6
  - Project demos during last week of class

- In-class exams:
  - Two in-class exams during week 6 and week 11

- NO LATE SUBMISSIONS OR EXAM RESCHEDULING WITHOUT PRIOR APPROVAL...IF YOU HAVE AN EMERGENCY CONTACT ME ASAP
Course Organization (cont’d)

- No required textbook
  - No single book or even half dozen books covers the content
  - Some content is so current no textbooks cover the material yet
  - Trade-off between timely publication and accuracy

- Will draw material from
  - Previous lecture material, links to articles, websites, datasheets
  - H&P Computer Architecture, P&H Computer Organization

- Will try to incorporate real world examples
  - Not always...but sometimes... the latest technology
  - Concepts sometimes more easily introduced and understood in context of older, simpler systems
    - Same techniques often used in more recent systems; In some cases the functionality is preserved intact
  - Studying both older and newer systems allows us to see and understand the evolution of features and reasoning
## Grading

<table>
<thead>
<tr>
<th>Points</th>
<th>Letter grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>≥ 94</td>
<td>A</td>
</tr>
<tr>
<td>90 – 93</td>
<td>A-</td>
</tr>
<tr>
<td>87 – 89</td>
<td>B+</td>
</tr>
<tr>
<td>84 – 86</td>
<td>B</td>
</tr>
<tr>
<td>80 – 83</td>
<td>B-</td>
</tr>
<tr>
<td>77 – 79</td>
<td>C+</td>
</tr>
<tr>
<td>74 – 76</td>
<td>C</td>
</tr>
<tr>
<td>70 – 73</td>
<td>C-</td>
</tr>
<tr>
<td>67 – 69</td>
<td>D+</td>
</tr>
<tr>
<td>64 – 67</td>
<td>D</td>
</tr>
<tr>
<td>60 – 63</td>
<td>D-</td>
</tr>
<tr>
<td>&lt; 60</td>
<td>F</td>
</tr>
</tbody>
</table>

I will pretty much follow the PSU grade guidelines to determine your final grades...with minor adjustments around the grade boundaries (usually to favor the students)

<table>
<thead>
<tr>
<th></th>
<th>ECE 485</th>
<th>ECE 585</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework</td>
<td>20%</td>
<td>20%</td>
</tr>
<tr>
<td>Exams</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Final project</td>
<td>30%</td>
<td>20%</td>
</tr>
<tr>
<td>Paper Summary</td>
<td>N/A</td>
<td>10%</td>
</tr>
</tbody>
</table>
Paper Summary

If you are taking the course as ECE 585, you are required to complete an additional assignment.

Twofold purpose:
- Give you an opportunity to explore a relevant and timely topic in some depth
- Acquaint you with a variety of useful sources; textbooks, journal papers, websites, user guides, etc.

Assignment:
- Choose a relevant topic and identify 3 good publications on the topic
- Read and summarize each publication (~1 page/each)
- Submit copies of the publications (or references if not available) and your summaries
Academic Honesty

- Students are expected to be honest in their academic dealings. Dishonesty is dealt with severely.
- Plagiarism occurs when you submit the work of any other person without acknowledging that it is not your own:
  - e.g. copying, with or without permission, homework, Verilog or program source code or exam answers from current or former student(s) in the class.
  - e.g. copying, with or without permission, homework, project reports, Verilog or program source code or exam answers from previous class postings, “course files” etc.
- If you are cutting and pasting text from another document, you are probably plagiarizing, unless you are quoting the material and citing the source.
- Read about PSU’s academic dishonesty policy and consequences at:
  
  http://www.pdx.edu/dos/conductcode
Academic Honesty (cont’d)

- The work of the team is treated as a unified or collective effort
- Contributions from non-team members may be considered academic dishonesty if they go beyond reasonable “help”
- The same standards for citation and attribution apply to team projects
  - If it’s not your original code and not provided by the instructors than acknowledge the source
  - If it’s not your original words in a write-up than acknowledge the source
- Team is expected to identify the individual contributions of team members in the write-up
- Team members are encouraged to provide confidential feedback which could affect an individual grade
PLAGIARISM PENALTY - At a minimum 0 points on the entire assignment for all parties involved

The millisecond you decide to submit work that is not your own you have decided that it is an acceptable risk:

- That you won’t receive an A or A- in the course
- That your partner(s) won’t receive an A or A- in the course
- That the classmate(s) you copied from won’t receive an A or A- in the course
- That you could lose your advisor, lose your TA or RA position, lose your financial aid, lose your ability to graduate, ...

☐ It is ECE department policy to report incidents of plagiarism to the PSU Administration for further action
Software Tools

- Verilog Simulators
  - Mentor Graphics:
    - You will receive a license file by email from Mentor Graphics after completing online registration form
  - SynaptiCAD Verilogger
    - Easy to use Verilog simulator
    - Free download from [http://www.syncad.com/verilogger_verilog_simulator.htm](http://www.syncad.com/verilogger_verilog_simulator.htm)
    - Limited, but adequate for this course. Can also purchase license
  - *Both simulators are available on the Windows PC’s in the Intel Lab in FAB on campus*

- C/C++ Compilers
  - Pick from many Windows (incl. Cygwin/GCC) and Linux
    - [http://cplus.about.com/od/glossary/a/compilers.htm](http://cplus.about.com/od/glossary/a/compilers.htm)
Course Outline

Posted on course website
Finite State Machines Review

Source: Lecture material and example provided by Mark F.
Ex: T-bird Tail Lights
Ex: State Transition Diagram

□ Inputs:
- Left (turn) off/on
- Right (turn) off/on
- Hazard Flasher off/on

□ Outputs:
- LC, LB, LA off/on
- RC, RB, RA off/on

□ States:
- IDLE – Wait for input asserted
- L1, L2, L3 – Control left tail light
- R1, R2, R3 – Control right tail light
- LR – Control both tail lights
Ex: State Encoding

- **Encoded**
  - 8 states
  - $2^3 = 8$
  - Need 3 flip flops
  - Need to determine state assignment

- **One-hot**
  - Dedicate a flip flop per state
  - Need 8 flip flops
  - Good choice for FPGA implementation. Why?
Ex: Binary Encoded Moore Machine
Ex: Binary Encoded FSM Output Logic

<table>
<thead>
<tr>
<th>State</th>
<th>LC</th>
<th>LB</th>
<th>LA</th>
<th>RA</th>
<th>RB</th>
<th>RC</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>R3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LR3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
LC &= Q_2' \cdot Q_1 \cdot Q_0' + Q_2 \cdot Q_1' \cdot Q_0' \\
LB &= Q_2' \cdot Q_1 \cdot Q_0 + Q_2' \cdot Q_1 \cdot Q_0' + Q_2 \cdot Q_1' \cdot Q_0' \\
LA &= Q_2' \cdot Q_1' \cdot Q_0 + Q_2' \cdot Q_1 \cdot Q_0 + Q_2' \cdot Q_1 \cdot Q_0' + Q_2 \cdot Q_1' \cdot Q_0' \\
RA &= Q_2 \cdot Q_1' \cdot Q_0 + Q_2 \cdot Q_1 \cdot Q_0 + Q_2 \cdot Q_1' \cdot Q_0' + Q_2 \cdot Q_1' \cdot Q_0' \\
RB &= Q_2 \cdot Q_1 \cdot Q_0 + Q_2 \cdot Q_1 \cdot Q_0' + Q_2 \cdot Q_1' \cdot Q_0' \\
RC &= Q_2 \cdot Q_1 \cdot Q_0' + Q_2 \cdot Q_1' \cdot Q_0'
\end{align*}
\]
Ex: Binary Encoded FSM Next State Logic

All cancel out
Ex: Binary Encoded Next State Logic (cont’d)

Q2* = Q2’ \cdot Q1’ \cdot Q0’ \cdot (HAZ + LEFT \cdot RIGHT) 
+ Q2’ \cdot Q1’ \cdot Q0’ \cdot (RIGHT \cdot HAZ’ \cdot LEFT’)
+ Q2’ \cdot Q1’ \cdot Q0’ \cdot (HAZ)
+ Q2 \cdot Q1 \cdot Q0 \cdot (HAZ'')
+ Q2 \cdot Q1’ \cdot Q0 \cdot (HAZ’)
+ Q2 \cdot Q1 \cdot Q0 \cdot (HAZ'''
+ Q2 \cdot Q1 \cdot Q0 \cdot (HAZ)

Q1* = Q2’ \cdot Q1’ \cdot Q0 \cdot (HAZ’)
+ Q2’ \cdot Q1 \cdot Q0 \cdot (HAZ'')
+ Q2 \cdot Q1’ \cdot Q0 \cdot (HAZ’)
+ Q2 \cdot Q1 \cdot Q0 \cdot (HAZ'''

Q0* = Q2’ \cdot Q1’ \cdot Q0’ \cdot (LEFT \cdot HAZ’ \cdot RIGHT’)
+ Q2’ \cdot Q1’ \cdot Q0 \cdot (RIGHT \cdot HAZ’ \cdot LEFT’)
+ Q2’ \cdot Q1’ \cdot Q0’ \cdot (HAZ’)
+ Q2 \cdot Q1’ \cdot Q0 \cdot (HAZ’')
Ex: One-Hot Encoded Next State Logic

\[
\begin{align*}
\text{IDLE}^* &= \text{IDLE} \times (\text{HAZ} + \text{LEFT} + \text{RIGHT})' + \text{L3} + \text{R3} + \text{LR3} \\
\text{L1}^* &= \text{IDLE} \times \text{LEFT} \times \text{HAZ}' \times \text{RIGHT}' \\
\text{R1}^* &= \text{IDLE} \times \text{RIGHT} \times \text{HAZ}' \times \text{LEFT}' \\
\text{L2}^* &= \text{L1} \times \text{HAZ}' \\
\text{R2}^* &= \text{R1} \times \text{HAZ}' \\
\text{L3}^* &= \text{L2} \times \text{HAZ}' \\
\text{R3}^* &= \text{R2} \times \text{HAZ}' \\
\text{LR3}^* &= \text{IDLE} \times (\text{HAZ} + \text{LEFT} \times \text{RIGHT}) + (\text{L1} + \text{L2} + \text{R1} + \text{R2}) \times \text{HAZ}
\end{align*}
\]
Ex: One-Hot Encoded Next State Logic (cont’d)

\[ LR3^* = \text{IDLE} \ast (\text{HAZ} + \text{LEFT} \ast \text{RIGHT}) + (\text{L1} + \text{L2} + \text{R1} + \text{R2}) \ast \text{HAZ} \]

**Gate Level Verilog:**

```verilog
wire v, w, x, y, z, LR3_star;

or o1 (v, HAZ, LEFT);
and a1 (w, v, RIGHT);
and a2 (x, IDLE, w);

or o2 (y, L1, L2, R1, R2);
and a3 (z, x, HAZ);

or o3 (LR3_star, x, z);
```

**Dataflow Level Verilog:**

```verilog
wire LR3_star;

assign LR3_star = (IDLE & (HAZ | LEFT & RIGHT)) | ((L1 | L2 | R1 | R2) & HAZ);
```
Example

...or better yet, write behavioral Verilog

...But will it really work?
Ex: Max Clock Speed
Ex: Max Clock Speed (cont’d)
...but the “real world” is even more complicated

- Even with careful routing the clock signal will not reach all of the flip-flops at exactly the same instant
  - This is called clock skew...and it needs to be factored into our max speed calculation
- Clock Period_{min} = FF t_{pd} + FF t_{setup} + CL t_{pd} + t_{skew}