**ECE 111**

*Instructor*: John Eldon

*Format*: Lectures in Person; Discussion Remote

*Online Teaching Platforms* : Zoom for discussions. Canvas for lecture slides, labs, final project, announcements. Piazza for Q&A, day-to-day interaction with instructor and TA’s. Gradescope to submit homeworks and final projects. Gradescope will be linked through class Canvas.

*Lectures*: With a few exceptions to be announced, all lectures will be delivered in person. Recordings of the lectures will be available on Canvas for asynchronous viewing. Lecture slides will be posted on Canvas.

*Discussions*: Each week prof and/or TA’s will conduct a one-hour discussion session to provide help and support to students on homeworks/labs using Zoom. Each TA will publish their in person and Zoom based office hours with a zoom link. Recordings of some of the important TA sessions will be available on Canvas for later offline viewing. Final project discussions will be conducted by the Instructor during lecture hours and TA’s will conduct additional sessions on final project discussion. Schedule for project discussion will be posted on Canvas

*Labs*: The required labs are remote and simulation-based. Instructions will be provided to students on how to download required software and tools to perform synthesis and simulation of code to complete each lab. Students should be able to download and/or access required software through their personal machines (windows or mac machine). Software download steps or steps to access software through UCSD's CloudLabs will be posted on class canvas. There will be a dedicated TA session in the first two weeks to help students set up simulation and synthesis software.You may also use the EDAPlayground for simulation and synthesis of your Verilog code.

*Homework*: Student will complete the homeworks (i.e. Labs) and submit the report in PDF format through Gradescope. Format of each homework report and code submission requirements will be provided to students. Each homework may include one or multiple Lab (design RTL code, synthesis and simulate). Lab will be small to medium size RTL code. Code framework for each lab will be provided. Students will be required to develop design code for each lab and for most homeworks testbench code will be provided by the instructor. Specific details on each Lab/Homework will be discussed during the class and these requirement details will be published on class canvas and piazza. Each week TA will provide assistance to students when executing homework labs. Students can ask questions in class piazza on homework labs. Instructor and TA will provide their feedback on the piazza. Each student will submit their individual homework/lab’s using a gradescope.

*Final Project*: Course will include the final project which will be due by the last day of the class. This would be a group project where students can have a team of 1 or 2 or 3 students. Final project discussion sessions and TA support will be provided to students through zoom sessions. Additionally, Students should be able to ask questions to both instructors and TA through piazza. Final project requirements, background of design requirement, concepts and guidance will be provided to students. Students can form a final project group on Canvas.

Class Outline

1. Intro

Introduction to terminology

* + SoC, FPGA, ASIC, HDL, Design, Synthesis, Simulator

Full custom chip designing process and challenges

Role of hardware description languages

SystemVerilog Capabilities and Syntax

Applications of FPGAs

2. ASIC, FPGA, Logic Synthesis

Introduction to ASIC and FPGA based prototyping

ASIC and FPGA design flow

Introduction to FPGA Architecture and its internal components

Logic Synthesis, Simulation, Simulator, Waveform

ASIC versus FPGA differences

3. SystemVerilog Modeling Abstraction

Combinational vs. Sequential

Behavioral vs. Register Transfer Level Data Flow Modeling

4. Anatomy of a SystemVerilog Module

specification format

ports, parameters

hierarchies

5. Data Types, Continuous Assignments, Conditional Operator

nets, wires, logics, regs, wireand (WAND)

synthesizable, non-synthesizable variables

signed (two’s comp.), unsigned

6. Blocking, nonblocking assignments

RTL event scheduling flow

inter-delay

shift registers, Johnson counters, barrel shifters

7. Procedural Blocks

initial, always

latch, flip flop, combinational

race conditions

LFSR / pseudonoise generator

8. RTL Programming Statements

9. Finite State Machines

Mealy v. Moore

state diagrams to Verilog

parity and error detection

UARTs and vending machines

10. Packed, unpacked arrays

memories

11. Test benches

tasks and functions

$stop and $finish

$write, $display, $strobe, $monitor