# Assignment Requirements for EE/CSE 371

## Technical Communication

**Effective technical communication** is a particularly critical skill once you graduate. This can take the form of writing research grants and papers, writing documentation for your work, teaching, or making presentations to your collaborators, managers, clients, or venture capitalists.

371 is primarily a lab-based course and we employ multiple forms of technical communication: **lab demos**, **lab reports**, **homework**, and **code commenting**. Lab demos allow you to practice *verbal* communication, while homework, lab reports, and code allow you to practice *written* communication.

The assignment documents (lab reports and homework), in particular, serve a couple of roles:

1) Communicate to us that you learned and/or achieved what we wanted you to and
2) Create a persistent digital artifact that you can refer back to in the future.

This last point is of particular importance should you ever need to look back on your work from this course as it pertains to your future academic, professional, or personal projects!

But just having a digital artifact is insufficient if you can't understand it! Along those lines, it is very important that your work is properly explained, including using good style in your SystemVerilog code. Having good style while you are writing code will also make debugging a lot easier!

In order to balance the importance of technical communication with the other work in this course, we only ask that the following requirements are met.

---

## Lab Report and Homework Requirements

- Lab reports and homework must be submitted as a *single* PDF file.
  1) You can print directly to PDF from Office products (like Microsoft Word) and combine multiple PDF files into one using Adobe Acrobat or online websites.
  2) Any materials *generated by hand* can be included by scanning or importing a photo.
  3) Any materials *generated electronically* can be imported or included via screenshot.

- SystemVerilog HDL files should be submitted separately and do not need to be included in your lab report and homework PDFs.

- Lab reports and homework should begin with your **name**, your **student ID number**, and the **assignment number** (*e.g.*, Lab 2, Homework 3).

- Everything included in your **lab reports** should be *comprehensible*, *properly labeled*, and *explained*. Someone who hasn't taken this class should be able to understand at a high level what you did and why just by reading your document.

- **Homework** will also involve explanations at times, but the formatting and detail requirements will be much laxer, as these are intended to be opportunities to practice with the course material.

# Lab Demo Notes

The lab demos are both a chance to show off your project live and to practice your verbal communication. You can also treat these as a chance to check in with the course staff and get some immediate feedback.

- The lab demos are not intended to be a secretive assessment; you should have a general idea of how you did by the end of every demo.
- If you worked with a partner, *both* partners must be present for the lab demo and are expected to *both* provide responses during the demo.

## Demonstrating Your Project

Depending on the lab, you may not be asked to demo every task. Pay special attention to the notes at the end of each lab spec to see generally what will be asked of you.

You will want to have your code already loaded into LabsLand and synthesized *before* your demo to save time. The first time, synthesis can take a long time but following iterations will be much faster, assuming no changes were made.

To show proper behavior of your system, you should already have some input sequences in mind (hint: refer back to your testbenches). For some labs, the TAs may ask you to demonstrate specific input combinations or behaviors, but these won't always be given ahead of time.

**Important:** The lab demo is *not* just about correctness; that will mainly be assessed in your lab report. The demo is about making sure that your project does what your report says it does and about your design process. You can still earn partial or even full points (in some cases) for the demo even if you implemented something incorrectly or didn't finish!

## Demo Questions

After the demonstration portion, the TA will ask you 1-2 questions about the lab. The demo questions generally fall into one of the following categories:

1) A review question based on lecture material related to the lab
2) Describe some piece of your design
3) Having you talk through a tweak to your design

These are *not* intended to be a grilling (*i.e.*, this is not an interview). These are more about your ability to talk about your design and design process. The TA may follow up with clarifying questions or feedback about your responses.

# Lab Report Outline

Not all of the following requirements will apply to every lab, but this outline and the rubrics at the end of the lab specifications are intended to act as a guide to your lab report writing and the lab report grading. If ever in doubt, please post any questions you have on the course discussion board.

Lab reports should include the following three main sections:

1) **Design Procedure**
   - ✓ Give a brief description of how you approached the tasks in this lab.
   - ✓ Include diagrams of your system design. This will always include a block diagram of your overall system but can also include finite state machine (FSM) and algorithmic state machine (ASM/ASMD) diagrams, depending on the lab.
   - ✓ For each major component of your system, (1) highlight important features and (2) discuss *how* and *why* you implemented it the way that you did.
     - Include snippets of important pieces of code if helpful to your discussion.

2) **Results**
   - ✓ Give a brief overview of the finished project, compared to what was asked for.
   - ✓ Include screenshots of your ModelSim results for important modules and *clearly describe what was tested and how the screenshots verify/demonstrate the desired behavior*.
     - If you are using a provided module or reusing a module from a previous lab, you do not need to verify its behavior, but you should mention where it came from.
   - ✓ Include a screenshot of the "Flow Summary" from your Compilation Report for your final Task. No explanation needed.

3) **Experience Report**
   - ✓ Give any positive or negative feedback you have on the lab. What was your implementation experience like?
   - ✓ Describe any significant issues you encountered while completing the lab and how you overcame them (or "I did not encounter any significant issues in this lab").
   - ✓ Include any tips and/or tricks that you discovered while completing the lab – these tips may prove useful to you in future labs!
   - ✓ Your lab report should end with your **estimated total time** (in hours) to complete the lab.
     - This includes reading, planning, design, coding, debugging, testing, etc.

See the separate document `Lab Report Example` to see a more concrete implementation of these different sections.

# Code Style Requirements

Your overall goal is to make your code easy to read and understand, particularly *without* having to dissect or even read every line of code. This will be helpful for others (and future you) to figure out what your code does, or is supposed to do, and how different parts of your code fit together.

- Choose meaningful and descriptive names for modules, parameters, wires, regs, logics, etc. If you do use vague names, make sure that there are comments that clarify their purpose.
- The contents of each block of code (*e.g.*, `always`, `initial`, `for`, `module`) should use an extra level of indentation.
  - `begin` can be on the same line as the start of the block or on the next line at the same indentation level.
  - Each `end` or `endmodule` should include a comment indicating which block it closes.
- Each module needs to have a block comment in front of it that
  1. Describes its overall function,
  2. Briefly explain the module ports, *including their bit-widths*, and
  3. Mentions any connections to other modules.
- Every *major* "chunk" of code (*e.g.*, blocks of code such as `always_comb`/`always_ff`, collections of `assign` statements) should have a comment explaining its purpose.
- Any unclear, messy, or complex lines of code should have an explanatory comment.

**Example:**

Each module should have a comment mentioning all inputs and outputs, as well as the bit-width of each (these are not necessary for `DE1_SoC`). The comment should also describe the module's functionality and other important notes.

```
1  // DE1-SoC is the top-level module for the full adder implemented in this lab. It
2  // adds the 1-bit values of the switches SW2 (A), SW1 (B), and SW0 (carry-in) and
3  // outputs the two-bit sum to the LEDs LEDR1 (carry-out) and LEDR0 (sum).
4  module DE1_SoC (HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, LEDR, SW);
5    input  logic [9:0] SW;
6    output logic [9:0] LEDR;
7    output logic [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
8
9    // Connect the signals, as specified in the header comment, to this fullAdder.
10   fullAdder FA (.A(SW[2]), .B(SW[1]),.cin(SW[0]), .sum(LEDR[0]), .cout(LEDR[1]));
11
12   // Drive HEX0 to HEX5 off
13   assign HEX0 = 7'b1111111;
14   assign HEX1 = 7'b1111111;
15   assign HEX2 = 7'b1111111;
16   assign HEX3 = 7'b1111111;
17   assign HEX4 = 7'b1111111;
18   assign HEX5 = 7'b1111111;
19
20 endmodule  // DE1_SoC
```

Every submodule instantiation should have a comment explaining the port connections and its role, particularly if this module type is instantiated more than once.

Large sections of code, including `always_comb` blocks, `always_ff` blocks, and multiple `assign` statements in a row should have a comment explaining their purpose and behavior.

Closing comment for modules should be the module name. Closing comment for long blocks can just be the block name (*e.g.*, `always_ff`). These comments make visually scanning your code much easier.