ECE 5745 Computer Architecture, Spring 2021
Lab Assignment Code and Report Logistics

`http://www.csl.cornell.edu/courses/ece5745`
School of Electrical and Computer Engineering
Cornell University

revision: 2021-03-05-10-44

This document describes what students are expected to submit for the lab assignment code and report and how their submissions will be evaluated. A lab handout is provided for each lab that describes: the motivation for the lab; provides background on the baseline design, alternative design, testing strategy, and evaluation; and discusses some details to cover in your lab report. The lab assignment is graded in two parts: the lab code and the lab report.

1. Lab Code

Completing the lab code will help students to achieve the following learning outcomes described in the course syllabus:

- **demonstrate** the ability to implement and verify designs of varying complexity at the register-transfer level and to push these designs through a commercial ASIC CAD flow.
- **create** new baseline and alternative designs at the register-transfer level, the associated effective testing strategies, and a thorough evaluation plan.

1.A Lab Code RTL Language

Students can choose to use either PyMTL or Verilog for their register-transfer-level modeling. All functional-level modeling, cycle-level modeling, verification, and simulator harnesses will be implemented in PyMTL. Students can also experiment with using both PyMTL and Verilog to complete a lab assignment, but they must choose one implementation to be graded. To choose which RTL language they want to use, students need to set the `rtl_language` variable in the top-level `RTL.py` file. After setting this variable, the test and simulation harnesses will automatically use the desired implementation. Students are free to delete the files associated with the RTL language they are not using to simplify their lab repository. For example, if students use PyMTL in the first lab, then they can delete the `VRTL.v` files. Students can change which language they use for each lab and the project.

1.B Lab Code Release

Initial code for each lab will be released throught GitHub, and students will be using GitHub for all development related to the lab assignments. Every lab group will have their own private repository as part of the `cornell-ece5745` GitHub organization, and all lab development must be done in this specific repository. **You should never fork your lab group's remote repository! If you need to work in isolation then use a branch within your lab group's remote repository.** The course instructors will merge new code into the each lab group's remote repository, and then students simply need to pull these updates.

1.C Lab Code Submission

The code will be submitted via GitHub. You just need to make sure that the final version of your code is pushed to your lab group's remote repository on GitHub before the deadline. Automated scripts will clone the `main` branch of each student's repository at 11:59pm on the due date, and then create an annotated tag to unambiguously denote what version of the code was collected. If you are trying to push last minute changes then it is likely our automated scripts may clone the wrong version. You should make sure your final code is pushed to GitHub at least five minutes before the deadline.

You should browse the source on GitHub to confirm that the code in the remote repository is indeed the correct version. Make sure all new source files are added, committed, and pushed to GitHub. You should not commit the `build` directory or any generated content (e.g., unit test outputs, VCD dumps, `.pyc` files). **Do not included any of the build directories used by the ASIC tools!** Including generated content in your submission will impact the grade for the assignment. **You should confirm that a clean clone of your lab assignment correctly builds and passes all of the tests you expect to pass using the following process:**

```
% mkdir -p ${HOME}/ece5745/submissions
% cd ${HOME}/ece5745/submissions
% git clone git@github.com:cornell-ece5745/labY-groupXX
% cd labY-groupXX
% mkdir -p sim/build
% cd sim/build
% pytest ..
% pytest .. --test-verilog
```

where `Y` is the lab number and `XX` is your group number. If, for any reasons, the above steps do not work, then you will not be able to score above a one on the RTL code quality criteria. For example, students occassionally forget to commit new source files they have created in which case these new files will not be in the remote repository on GitHub. We also must be able to run any simulators associated with the lab. Note, all the tests do not have to pass, but these steps must work so that we can easily build, test, and evaluate your code.

We will be using TravisCI to (partially) grade the code functionality for the lab assignments. So in addition to verifying that a clean clone works on the `ecelinux` machines, you should also verify that all of the tests you expect to pass are passing on TravisCI by visiting the TravisCI page for your lab repository:

- `https://travis-ci.com/github/cornell-ece5745/labY-groupXX`

where `Y` is the lab number and `XX` is your group number. If your lab is failing tests on TravisCI, then the score for code functionality will be reduced. Keep in mind that in the final few hours before the deadline, the TravisCI work queue can easily fill up. You should always make sure your tests are passing on the `ecelinux` servers and not rely solely on TravisCI to verify which tests are passing and failing.

2. Lab Report

Completing the lab report will help students to achieve the following learning outcomes described in the course syllabus:

- **write** comprehensive technical reports that describe designs implemented at the register-transfer level and evaluate the designs to determine the superior approach.

2.A Lab Report Structure

The lab report should be written assuming the reader is familiar with the lecture material and has read the lab handout. All lab reports should include a title, the names of the students in the group, and the NetIDs of the students in the group at the top of the first page. Do not put this information on a separate title page. You should include the following sections:

- **Section 1. Introduction (1 paragraph maximum) –** Students must summarize the purpose of the lab. Why are we doing this lab? How does it connect to the lecture material? There are often many purposes. Think critically about how the lab fits into the other labs. Students can paraphrase from the handout as necessary. Students must include a sentence or two that describes at a very high-level their alternative design. The introduction should be brief but still provide a good summary of the lab assignment.

- **Section 2. Alternative Design –** Students must describe their alternative design and their implementation. Students must include a datapath diagram or a block diagram and possibly an FSM diagram for the alternative design. Consider a paragraph that provides an overview of your design, before doing a deep dive into the details of one or two interesting aspects of the design. Think critically about what are the key items to mention in order for the reader to understand how the alternative design works. Examples are usually great to include here to illustrate how the alternative design works. Do not include waveforms. Do not include detailed information about PyMTL/Verilog signals or code; your lab report should be at a higher level. If you include line traces then you must annotate them so that the reader can understand what they mean. **Remember that you must provide a balanced discussion between what you implemented *and* why you chose that implementation.**

- **Section 3. Evaluation –** Students must report their simulation results *and* the area, energy, timing results from the ASIC flow using an appropriate mix of text, tables, and plots. Do not simply include the raw data. You must include some kind of summary; a plot is almost always helpful. You must include some kind of analysis of the results: Why is one design better or worse than another? Can you predict how the results might change for other designs or parameters? What can we learn from these results? Students must explicitly discuss the area and cycle time for each design and the performance (both in number of cycles and in nanoseconds) and energy for each input dataset. Describe where the critical path is located in your designs; you may want to draw the critical path on a datapath diagram. Be specific and explain *why* you think one design has a longer cycle time or uses more area or energy. Dig into the reports, do not just copy and paste the summary results from the automated standard-based ASIC flow as your evaluation. **You must include at least one amoeba plot of your alternative design! Remember to provide a balanced discussion between what the results are *and* what those results mean.**

- **Section 4. Conclusion (1 paragraph maximum) –** Students must include a brief qualitative *and* quantitative overview of the evaluation results in terms of the area, energy, cycle time, and execution time in cycles. Which design did best? By how much? On which inputs? Students must include some high-level conclusions they can draw from their quantitative evaluation. Do not over-generalize. Can you predict how the results might change for other inputs? What can we learn from these results? Which implemenation should we use in the future. If it depends, explain why it depends.

Many students initially struggle with the idea of preparing the PA report. In previous courses, students often simply describe their code at a low level in a lab report. In this course, we are challenging students to prepare reports that better demonstrate the student's understanding of the course content. Before starting to write the report, we encourage students to prepare a detailed outline. The outline should include one section for each of the four sections that will eventually make up the report. Under each section, there should be one bullet for each paragraph the student is planning to include in that section. This bullet should describe the topic of the paragraph. Under each bullet there should be several sub-bullets, one for each topic to be discussed in that paragraph. The outline should also explicitly include references to the figures, tables, and plots the student plans to include in the report. This is called a structured approach to technical writing. Students are strongly discouraged from "just starting to write". Just like we should always plan our approach before starting to implement hardware, we should plan our approach before writing the report.

It is also always great to include extra material to help demonstrate your understanding. For example, you could include line traces and reference them to illustrate a key feature of your design or to illustrate a subtle bug, or reference them in your evaluation to illustrate *why* a specific input pattern performs the way it does. If you include line traces you must annotate them. Label the columns and maybe even draw on them to show what is going on. Including waveforms is usually not helpful, and including a bunch of code is usually not helpful. You could maybe include a particularly clever test case. You could include a pen-and-paper example to illustrate how alternative design works. Also be sure to highlight "extra" work you did in your design, testing, or evaluation. If you tried two different alternative designs discuss them in the alternative design section and make sure to use them to create a richer comparative analysis in the evaluation section. If you added an interesting new evaluation input, make sure you highlight that in your evaluation section. You can explain any new ways you found to use the ASIC tools to provide deeper insight. There are many creative things you can do to set your report apart!

We recommend students use Google docs to collaborate on the lab report. For example, you can create a Google doc to track ideas and brainstorming. You can upload diagrams and such so everyone has a copy. Instead of emailing documents, just work collaboratively in Google docs. You can always see the latest version, it is backed up in the cloud, and it is simple for multiple students to be writing on different parts of the document at the same time. It is also trivial to export to PDF, or you can cut-and-paste a near-final version into a different word-processor for final formatting.

2.B Lab Report Formatting

The lab report should be written using a serif font (e.g., Times, Palatino), use margins in the range of 0.5–1 in, and use a 10 pt font size. Please choose a line spacing and margins so that your report is readable. All figures must be legible. Avoid scanning hand-written figures and **definitely do not use a digital camera to capture a hand-written figure**; the lab report is too important to risk an illegible figure. Clearly mark each section with a *numbered* section header. Your report can be a maximum of six pages.

2.C Lab Report Submission

Your report should be uploaded to Canvas.

3. Lab Assessment Rubric

The lab code and report will be graded using the following breakdown:

• Code: Functionality	65%
• Code: Code Quality	5%
• Lab Report: Introduction/Conclusion	5%
• Lab Report: Alternative Design	10%
• Lab Report: Evaluation	10%
• Lab Report: Writing Quality	5%

Each criteria is scored on a scale from 0 (nothing) to 4.25 (exceptional work). In general, a score of 3 is awarded for reasonable work while a score of 4 is reserved for very strong work. The functionality of the alternative design models are assessed based on the number of test cases that pass in the test suite and whether or not the designs were able to be pushed through the standard-cell ASIC flow. The code quality is based on: how well the code follows the course coding guidelines (see the cheat sheet); inclusion of comments that clearly document the structure, interfaces, and implementation of all modules; following the naming convention and build system structure appropriately; decomposing complicated monolithic expressions into smaller sub-expressions to increase readability; cleanly separating combinational and sequential logic; using local parameters for constants; organizing the code logically to match the dataflow in the design. Overall, good code quality means little work is necessary to figure out how the code works and how we might improve or maintain the design.

4. Lab Academic Integrity

Students are explicitly prohibited from sharing their code with anyone that is not within their group or on the course staff. This includes making public forks or duplicating this repository on a different repository hosting service. Students are also explicitly prohibited from manipulating the Git history or changing any of the tags that are created by the course staff. The course staff maintain a copy of all repositories, so we will easily discover if a student manipulates a repository in some inappropriate way. Normal users will never have an issue, but advanced users have been warned.

Students are explicitly prohibited from reviewing or copying from lab reports from prior offerings of the course. The submitted lab report should be independently developed and accurately represent the students' understanding of the lab assignment.

Sharing code, manipulating the Git history, changing staff tags, or copying from prior lab reports will be considered a violation of the Code of Academic Integrity. A primary hearing will be held, and if found guilty, students will face a serious penalty on their grade for this course. More information about the Code of Academic Integrity can be found here:

- `http://www.theuniversityfaculty.cornell.edu/AcadInteg`