This document describes what students are expected to submit for the lab assignments and how their submissions will be evaluated. A lab handout is provided for each lab that describes the motivation for the lab and provides background on the baseline design, alternative design, testing strategy, and evaluation.

1. Lab 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.

2. Lab Code Release

Initial code for each lab will be released through 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.

3. 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 `master` 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:**

```bash
% 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
% py.test ..
% py.test .. --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 occasionally 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:

```plaintext
• https://magnum.travis-ci.com/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 ecelinux and not rely solely on TravisCI to verify which tests are passing and failing.

### 4. Lab Report Submission

The lab report should be written assuming the reader is familiar with the lecture material, but do not assume that the reader has read the lab handout; thus you might need to paraphrase some of the content in the lab handout in your own words to demonstrate understanding. Details about the actual code should be in the code comments. The lab report should focus on the high-level design aspects of the lab assignment. 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. Please choose the font size, line spacing, and margins so that your report is readable. Clearly mark each section with a numbered section header. You should include the following sections:
• **Section 1. Introduction** – 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 must describe their progress on the lab (Did you complete the alternative design? Were you able to push everything through the ASIC toolflow?). Students must include a sentence or two that describes at a very high-level their alternative design. 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. The introduction should be brief (0.50–0.75 pages) but still provide a good summary of the lab assignment.

• **Section 2. Baseline Design** – Students must briefly describe the baseline designs. Students must include a datapath diagram or a block diagram and possibly an FSM diagram for the baseline design, even though this might simply include copying these diagrams from the lab handout. Students must explain how the baseline design works; think critically about what are the key items to mention in order for the reader to understand how the baseline design works. This will probably be a relatively short section (0.50–0.75 pages) for this course since the baseline designs are provided for you.

• **Section 3. 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. Students are strongly encouraged to describe how their design incorporates specific design patterns and principles discussed in lecture and the discussion section, but be specific. Simply saying the design exhibits modularity, hierarchy, encapsulation, regularity, and/or extensibility is not sufficient; be specific and explain how the design exhibits these design principles. 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. **If a CL model is required for the lab, then you should explain your CL model here in the alternative design section.** This section will likely be about one page. **Remember that you must provide a balanced discussion between what you implemented and why you chose that implementation.**

• **Section 4. Testing Strategy** – Students must describe the testing framework provided for testing your design. Students must describe the overall testing strategy (e.g., unit testing, directed tests, random tests, whitebox vs. blackbox testing, assertion-based testing). Simply saying the group used unit testing is not sufficient; be specific and explain why you used a specific testing strategy. Students must explain at a high-level the kind of directed tests cases they used and why they used these test cases. We recommend students start this section with a short paragraph that provides an overview of your strategy for testing (so how all of the testing fits together). Then you might have one paragraph for each kind of testing. Each paragraph starts with the “why” (why that kind of testing) and then goes on to the “what” (what did you actually test using that kind of testing). Even though in this course most of the tests are provided for you, students are still trying to provide a compelling, evidence-based argument that there design is functionally correct. Do not include waveforms. Do not
include the actual test code itself; 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 (i.e., what corner case does the line trace illustrate?). This section will likely be 0.75 pages. **Remember to provide a balanced discussion between how you tested your design and why you used that testing strategy and test cases.**

- **Section 5. Evaluation** – Students must report their simulation results and the area, energy, timing results from the ASIC toolflow 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 each design; 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 use the summary.txt file for your evaluation. **You must include at least one amoeba plot of your alternative design!** There is no conclusion, so the big picture summary should really be in the evaluation. This section will probably be one of the longer (and most important) sections. **Remember to provide a balanced discussion between what the results are and what those results mean.**

We cannot stress enough the importance of the above description of our expectations. Every year many groups simply do not read the above description close enough. These groups do not include datapath/FSM diagrams, do not explain why they chose a specific testing strategy, do not include any analysis of cycle time, area, energy, etc. Please read the above description of our expectations closely. **If we say you must include something, then you must include it!**

It is also always great to include extra material to help demonstrate your understanding. For example, you could include line traces and reference them in the alternative design to illustrate a key feature of your design, or reference them in the testing strategy section to illustrate a subtle bug or a kind of testing, 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 include a bunch of code is usually not helpful. You could include a particularly clever test case and reference it in the testing strategy section. You could include a pen-and-paper example to illustrate how your baseline design or 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 used a new kind of testing technique (e.g., randomly generating different mixed instruction sequences) then make sure you highlight that in the testing strategy. 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!

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. 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.

Sections 1–5 (including the title and author list) can be a maximum of four pages. We do not
recommend including diagrams, plots, and tables throughout your discussion since this means you will have less room for text (and puts pressure on making the diagrams, plots, and tables too small). Instead, you can include as many pages at the end of your report with just the diagrams, plots, and tables. Be sure to number your diagrams, plots, and tables and reference them throughout your discussion.

We highly recommend students use Google docs to collaborate on the lab assignment. For example, you can create a Google doc to track ideas and brainstorming. You can upload diagrams and such so everyone has a copy. You can also create a Google spreadsheet to create an initial project roadmap and to track your progress. Perhaps most importantly, Google docs is a great way to collaborate on the final report document. 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.

5. Assessment Rubric

The rubric includes the following nine criteria most of which are weighted equally except for the lab report introduction section, the lab report writing quality, and the code quality which are weighted half as much as the other criteria.

- (×2) Code: Functionality of Alternative Design CL Model
- (×2) Code: Functionality of Alternative Design RTL Model
- (×1) Code: Code Quality
- (×1) Lab Report: Introduction
- (×2) Lab Report: Baseline Design
- (×2) Lab Report: Alternative Design
- (×2) Lab Report: Testing Strategy
- (×2) Lab Report: Evaluation
- (×1) Lab Report: Writing Quality

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. 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.

Please note that the assessment rubric places more weight on the lab report (approximately 66%) than simply writing the models and getting them to function correctly (approximately 33%). This is because the lab report is a much better indicator of a student’s understanding of the material. **Students should not be surprised if they receive an overall score of 3.0–3.5 for a working lab assignment with a reasonable lab report that mostly describes what they implemented, tested, and evaluated without any meaningful insight into why they chose a specific implementation, why
they chose a specific testing strategy, and what the results mean. Students should budget their time and resources appropriately. Intentionally describing functionality in the lab report which is not present in the RTL code will be considered a violation of the academic integrity code.

6. **GitHub and Academic Integrity Violations**

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.

Sharing code, manipulating the Git history, or changing staff tags 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](http://www.theuniversityfaculty.cornell.edu/AcadInteg)