ECE 4750 Computer Architecture, Fall 2016
Lab Assignment Assessment Rubric

`http://www.csl.cornell.edu/courses/ece5950`
School of Electrical and Computer Engineering
Cornell University

revision: 2016-09-28-11-38

This document describes the lab assignment roles, 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 Assignment Roles

Students are expected to work in a group of three students for each lab assignment, although groups of two or four students may be allowed with explicit instructor in exceptional circumstances. It is suggested that students form a group early on and keep the same group throughout the semester, although changing groups is allowed if necessary. For each lab assignment, each student will take on one of two specific roles: *RTL design engineer* and *RTL verification engineer*. Each role is described in more detail below.

- **RTL Design Engineer –** The RTL design engineers are responsible for the RTL implementation of the architecture. The RTL design engineers will likely start working on implementing the baseline design very early on in the project, and then move onto the alternative design later in the project. The RTL design engineers will need to work closely with the RTL verification engineers to ensure that directed tests specifically targeting the baseline and alternative design achieve good test coverage. It is common for the RTL design engineers to also spend time writing test cases, and RTL design engineers will certainly be heavily involved with the RTL verification engineers in debugging failing test cases.
- **RTL Verification Engineer –** The RTL verification engineers are responsible for the verification of the RTL implementation. The RTL verification engineers will likely start working on initial test cases using the functional-level model provided in the lab harness very early on in the project, and then move onto directed and random unit testing as well as integration testing for the baseline and alternative design. It is common for the RTL verification engineers to make smaller contributions towards RTL implementation, and RTL verification engineers will certainly be heavily involved with the RTL design engineers in debugging failing test cases.

Note that there are three students per group and only two roles, so for a given lab assignment a group needs to assign either two RTL design engineers and one RTL verification engineer, or two RTL verification engineers and one RTL design engineer. Practically speaking, almost all groups assign two RTL design engineers and one RTL verification engineer. Note that regardless of roles, all students should contribute to writing the lab report.

In addition to the above two roles, the group must also select which student will serve as the *architect* for each lab assignment. Note that the position of architect is in *addition* to the standard roles of RTL design engineer and RTL verification engineer. The architect has extra duties which include: establishing the initial project roadmap, developing the architecture for the alternative design (e.g., sketching out a datapath diagram, FSM diagram, or block diagram; preparing some pseudocode), developing the high-level testing strategy, and developing the plan for evaluating the baseline and alternative designs. The architect should keep track of the lab's progress and whether or not the group is meeting its initial milestones. Essentially, the architect acts as the group leader for that lab assignment.

The group should decide which student will take on which role and who will be the architect during their first meeting to discuss the lab assignment. There is an explicit table in the README of each group's Git repo where students are required to enter in which student takes on which role and which student is the architect for which lab assignment. *Students must take on each role at least once over the course of labs 2–4. Each student must serve as the architect at least once over the course of labs 2–4.* So if a student is the architect for lab 2, then that student cannot be the architect for lab 3 or 4. There are no restrictions on roles/architect for lab 1 and 5.

The lab report should include a full-page *role and task* table with one column per student. The table should include a row that indicates the role of each student and which student was the architect. The table should also include a detailed list of the tasks completed by each student. Please be specific and include pen-and-paper design work, RTL implementation of a specific module, writing specific test cases, debugging a specific bug, running specific simulations, writing a specific section of the report, etc. We expect this table to fill a full page and to accurately capture the work contributed by each group member. The scope and detail of this table will impact the grade for the lab assignment.

2. Lab RTL Language

Students can choose to use either PyMTL or Verilog for their register-transfer-level modeling. All functional-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. For example, in the first lab assignment students will need to set the `rtl_language` variable appropriately in both the `IntMulBaseRTL.py` file and the `IntMulAltRTL.py` files. 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. While students can change which language they use for each lab, there are some subtle issues with using a different language for lab 1 and 2 which will require discussion with the course instructors.

3. Lab RTL Code Release

Initial RTL 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-ece4750` 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.

4. Lab RTL Code Submission

The RTL 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). 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}/ece4750/submissions
% cd ${HOME}/ece4750/submissions
% git clone git@github.com:cornell-ece4750/lab-groupXX <labname>
% cd <labname>
% mkdir -p sim/build
% cd sim/build
% py.test ../<labname>
```

where `XX` is your group number and `<labname>` is either `lab1_imul`, `lab2_proc`, `lab3_mem`, `lab4_net`, or `lab5_mcore`. 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 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://magnum.travis-ci.com/cornell-ece4750/lab-groupXX`

where `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` machines and not rely solely on TravisCI to verify which tests are passing and failing.

5. Lab RTL Revision

We will be incorporating a new aspect into the lab submission process this year. After the deadline, the course instructors will branch your submission and create a pull request on GitHub. The instructors will then commit the instructor tests and evaluation simulator into this pull request which will trigger a TravisCI build. This will enable the instructors and the students to immediately see how their lab submission does on both the student tests, the instructor tests, and the instructor evaluation inputs. If the student's lab fails some of the instructor tests, then the students are free to fix bugs and commit these changes as part of the pull request. The students are encouraged to add comments to the pull request indicating what they had to change to pass the instructor tests, and why the student tests did not catch this bug. This RTL revision *will not* mitigate a reduction in the code functionality score due to failing instructor tests, but it *will* enable the course staff to judge how severe a penalty to access. If it turns out that after the students fix a very small mistake in their code, their lab now passes all of the tests then this will result in a small penalty. If it turns out that the students have to fix a major mistake, then this will result in a larger penalty, but at least the students will have figured out what is wrong so that they can use this code in future lab assignments. Such lab revisions will need to be made within a few days of the deadline.

6. 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 architecture, design, and verification 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. There are no formatting restrictions but 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 baseline? the design alternative?). 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 number of cycles executed (Which design did best? By how much? On which inputs?). Students must briefly discuss the impact of the alternative design on cycle time (i.e., clock frequency), area, and energy. Students must include some high-level conclusions they can draw from their qualitative and 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 describe the baseline design and your implementation. 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. Examples are usually great to include here to illustrate how the baseline design works. Students should describe in detail any changes you made from the design described in the handout and why you made these changes. Students must explain why is this design a good baseline for comparison. Students are 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 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. This section will likely be less than a page.

- **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 baseline design works. Students are 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 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. 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 value tests, random delay 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 implemented and why they used these test cases. Consider including a table with a test case summary, or some kind of quantitative summary of the number of test cases that are passing. Students are trying to provide a compelling, evidence-based argument that there design is functionally correct. 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). Then you can end with a paragraph that pulls it all together and tries to make a compelling case for why you believe your 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 about one page. **Remember to provide a balanced discussion between how you tested your design *and* why you chose that testing strategy and test cases.**

- **Section 5. Evaluation –** Students must report their simulation results 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 include some kind of qualitative analysis of the impact of the alternative design on cycle time (i.e., clock frequency), area, and energy. Simply saying one design uses more area or energy is not sufficient; be specific and explain *why* one design might use more area or energy. There is no conclusion, so the big picture summary should really be in the evaluation. evaluation." We recommend students start with the performance analysis and then have three short paragraphs: one each on area, energy, and cycle time. This is where your qualitative analysis comes in on these important metrics. Then have a final short paragraph which provides a bit more detailed of the high-level conclusion you used at the end of your introduction. 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 qualitative 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. 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 you report with just the diagrams, plots, and tables. Your role and task table will be included at the end and does not count towards your page limit. 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.

7. Lab Assessment Rubric

The rubric includes the following 10 criteria most of which are weighted equally except for the lab report introduction section, the lab report writing quality, and the RTL code quality which are weighted half as much as the other criteria.

- RTL Code: Baseline Functionality ($\times 2$)
- RTL Code: Alternative Functionality ($\times 2$)
- RTL Code: Verification Quality ($\times 2$)
- RTL Code: Code Quality ($\times 1$)
- Lab Report: Introduction ($\times 1$)
- Lab Report: Baseline Design ($\times 2$)
- Lab Report: Alternative Design ($\times 2$)
- Lab Report: Testing Strategy ($\times 2$)
- Lab Report: Evaluation ($\times 2$)
- Lab Report: Writing Quality ($\times 1$)

Each criteria is scored on a scale from 0 (nothing) to 4.25 (exceptional work). In general, a score of 3 (B) is awarded for reasonable work while a score of 4 (A) is reserved for very strong work. Scores of 4.25 are relatively rare. The baseline and alternative functionality are assessed based on the number of test cases that pass in both the student and instructor test suites in combination with the severity of any errors. The verification quality is based on the judgment of the instructor in terms of how well the students' test cases actually test the design. 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 60%) than simply writing the RTL and getting it to function correctly (approximately 40%). 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.

8. 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`