UCLA EE 201A -- VLSI Design Automation – Winter 2026
Lab 2: Logic Synthesis with Cadence Genus

TA: Dedeepyo Ray

****** IMPORTANT ******

- Run your experiments and do your work on SEASnet server eeapps.seas.ucla.edu
  - You are responsible for backing up your code & data.
- All necessary files can be found in `/w/class.1/ee/ee201o/ee201ot2/2024_labs/lab2`
- **See last page for important submission instructions**
- **Please also upload your report to Gradescope**
- **Due Date: Tuesday, Feb 3, 2026 at 11:59:59 P.M. Pacific Daylight Time (PDT)**

Notes:

- *You will find it useful to refer to the Cadence Genus tutorial from class and the official Genus user manual.*
- *Remember to set up your shell environment using* `new_csh_ee201a_setup` *(Additional commands added compared to the one used for Lab1).*
- *The only file you can modify to answer the questions in this assignment is the provided skeleton Tcl script. You may not change the input design itself (Verilog file), or the Nangate library (Liberty file) for any part of the lab.*
- *Please use Piazza for asking (and answering) questions before coming to office hours. However, you are expected to work on the assignment alone, so do not provide solutions or compare results with your peers.*

Purpose

For this assignment, you will perform synthesis of a small circuit (ISCAS89 s15850) with Cadence Genus Synthesis Solution using Nangate 45nm library. The objective of this lab is to give a hands-on experience with a synthesis tool on a real design in a recent technology.

For this lab, you should start with the Genus Tutorial slides shared earlier. To answer the questions in the lab, you should familiarize yourself with more advanced Genus functionality by reading its documentation, including the user manual (pointer available in the Tutorial slides). You can also use the tool help on the command line or in GUI mode.

You will be provided with all the materials to get started with the assignment. You will take the RTL description of the design and perform elaboration and synthesis. To perform synthesis targeting an ASIC, you basically need:

- RTL (or behavioral) descriptions of the design in Verilog
- Technology and user-specific libraries
- Constraints (to meet chip design objectives, such as timing)
- … and of course, a synthesis tool.

Electronic design automation (EDA) tools like Genus are often scripted by designers to perform many specific steps in sequence. The most common language for doing this in industry is the Tool Command Language, or Tcl. To answer the questions in this lab, you will be modifying the skeleton Tcl script that is provided.

Background

As discussed in class, RTL descriptions of the design specify the logic/control flow descriptions of the hardware. These descriptions can be written in a behavioral style (e.g., c = a+b implies the behavior of an adder block) or a structural style (e.g., modules and wiring connections that together compose an adder). You may take a look at the RTL descriptions in s15850.v to understand what a structural RTL representation looks like.

Technology libraries provide timing/power/functional descriptions of logic blocks that form the basis for building designs. Synthesis tools use the information from these libraries to map design instances in a generic netlist (obtained after elaboration phase) to technology-specific cells.

Timing constraints describe clock characteristics (e.g., period, slew rate, jitter, skew, etc.) and required input/output delays of ports/pins in the design. Timing constraints are usually described in tool-proprietary formats (i.e., every synthesis tool has a different form of constraint specification). However, one of the most popular and widely-used format is the Synopsys Design Constraint (SDC) format. Most tools can read in SDC format, including Genus.

Assignment

Problem 1 (3 points): TCL practice problem
This problem is designed for you to get familiar with Tcl scripting, which you will use a lot in Genus and Innovus (later). Genus or Innovus is not involved in this problem, as it's a pure Tcl practice. You should be able to read basic files and perform basic numerical operations after this problem.
You will be given a file 'wirelength.txt', which contains the wirelength data fetched from a design (each line contains one wirelength). Your goal is to write your own Tcl script to read this file, and use the data to answer the following questions:

A. Average wirelength
B. Number of wires with length larger than 50

In your report, list your results and **attach the screenshot** of your Tcl script. To run a Tcl script, use

```
$ tclsh xxx.tcl
```

Useful Tcl tutorials:
https://www.tutorialspoint.com/tcl-tk/index.htm
https://tcl.tk/man/tcl8.5/tutorial/tcltutorial.html

Problem 2 (5 points): Find the fastest achievable operating clock frequency for the design with the given technology library.

Part A (2 points) – Using a copy of the provided Tcl script, find the best clock period that can be achieved without creating timing violations. To do this, you will need to synthesize the design with different target clock period constraints. Repeat until you minimize the slack of the critical path.

Note: First fill out the missing parts in the skeleton script (parts marked with ???) and set your base working directory. Then set the clock period. Negative slack is a timing violation.

Report Deliverables: Include the fastest clock period you achieved. You may find that a clock period of X violates timing, while 0.97X passes timing! Please explain why this can happen.

Part B (3 points) – Improve the attainable clock speed even more by adding, modifying, or removing the commands in a ***new*** copy of the skeleton Tcl script.

Note: You may not change the timing constraints other than the clock period.

Hint: Try optimizing "total negative slack" and synthesize the design using different "effort" level.

Report Deliverables: Include the fastest clock period you achieved with this version of your Tcl script. Please list exactly which commands you added, modified, or removed from the skeleton script. Also explain why you chose those commands and what you hoped they would achieve.

Grading Notes: 1 point for trying and the answer in the report, even if you did not improve on results from Part A. 1 point if you get any improvement over part A. 1 point for the student(s) who achieve the fastest clock with no timing violations.

Problem 3 (5 points): Area vs. delay tradeoff and power optimization.

Part A (2 points) – Starting from your Tcl script used for Problem 2B, run your synthesis flow while increasing the clock period constraint from the smallest period you achieved in Problem 1B until the slack in the most critical path is no more than 200 ps.

Report Deliverables: Plot the curve of total synthesized design area versus clock period constraint (area on Y-axis and delay on X-axis) for at least 8 clock period values. What trend do you notice? Can you explain it?

Part B (3 points) – Starting from the skeleton Tcl script (after replacing the ??? with correct file/design names), modify it to synthesize the design and optimize for low power without violating timing constraints.

Note: You may not change timing constraints. Clock period must be 1000 ps and slew rate must be 100 ps as shown in the skeleton Tcl script.

Hint: Try constraining the max dynamic power and/or leakage power. Also try "retiming" for minimum area.

Report Deliverables: For the lowest total power you achieved with this version of your Tcl script, report the leakage/dynamic/total power breakdown. Please list exactly which commands you added, modified, or removed from the skeleton script. Also explain why you chose those commands and what you hoped they would achieve.

Grading Notes: 1 point for trying and the answer in the report. 1 point if you get any improvement in total power over the skeleton Tcl script. 1 point for the student(s) who achieves the lowest power.

<u>Submission Instructions (read carefully)</u>

- All necessary code, data, and report files must be tarballed and submitted as a single archive file on eeapps.
  - For final submission, create a directory named as follows: `Lastname-Firstname_UID_Username_Lab2/`
  - Inside this submission directory, you must include exactly seven files:
    - `Lastname-Firstname_UID_Username_results_submission.txt`
      - (The filled-in results_submission.txt, for autograding. IMPORTANT!)
    - `Lastname-Firstname_UID_Username_Lab2_2B.tcl`
      - (synthesis script for Problem 2B)
    - `Lastname-Firstname_UID_Username_Lab2_2B.v`
      - (synthesized Verilog output by Genus for Problem 2B)
    - `Lastname-Firstname_UID_Username_Lab2_3B.tcl`
      - (synthesis script for Problem 3B)
    - `Lastname-Firstname_UID_Username_Lab2_3B.v`
      - (synthesized Verilog output by Genus for Problem 3B)
    - `Lastname-Firstname_UID_Username_Lab2_1.tcl`
      - (Tcl script for Problem 1)
    - `Lastname-Firstname_UID_Username_Lab2.pdf`
      - (lab report)

  - Compress and archive this directory using tar/gzip to have a single submission file named `Lastname-Firstname_UID_Username_Lab2_pinXXXX.tar.gz` (Make up a 4-digit numeric PIN of your choice to substitute for XXXX – need not be the same as Lab 1)
  - Submit tarball by copying it to `/w/class.1/ee/ee201o/ee201ot2/submission/lab2`
    - IMPORTANT: Make sure all files you submit, as well as the tarball, have full read and execute permissions to group and others or we will not be able to grade your lab. Do this using `chmod -R go+rx FILE_OR_DIRECTORY`
  - **Late submissions will not be accepted** (or possible – read/write/execute permissions to `submission/lab2/` will be disabled at the deadline). If you cannot finish the entire assignment on time, submit whatever you completed. **No submission = no points.** You are welcome to overwrite your submission as many times as you like before the deadline. However, please only use the filename format that is provided. Duplicates of the form "v1, v2, v3, new, newest," etc. will be ignored. If you accidentally submitted your tarball multiple times using different PINs, only the latest timestamp will be considered.
  - Some students may worry that their work could be visible to other students. We try to reduce this chance by disabling read permissions on `/w/class/ee201o/ee201ot2/submission/lab2/` directory so that other students cannot list its contents. Thus, they will not know the filename and cannot open your submission unless you give them your full name, student ID number, SEAS username, and arbitrary 4-digit PIN that you substituted for `XXXX` earlier on the tarball. This also means you will not be able to list the directory contents to see if your own submission worked – you can only write to it! In short, please do not give your classmates this

information or collaborate on homeworks. The PIN can be whatever 4-digit number you want -- it is just to reduce the likelihood of another student guessing your full file submission path. You can change it for every lab submission if you like.

Submission Example

```
$ cd /w/class/ee201a/mgottscho
$ mkdir Gottscho-Mark_203555232_gottscho_Lab2
$ cp synth1b.tcl Gottscho-Mark_203555232_gottscho_Lab2/Gottscho-
  Mark_203555232_gottscho_Lab2_1B.tcl
$ cp synth2b.tcl Gottscho-Mark_203555232_gottscho_Lab2/Gottscho-
  Mark_203555232_gottscho_Lab2_2B.tcl
$ cp output/s15850_synth1b.v
  Gottscho-Mark_203555232_gottscho_Lab2/Gottscho-
  Mark_203555232_gottscho_Lab2_1B.v
$ cp output/s15850_synth2b.v
  Gottscho-Mark_203555232_gottscho_Lab2/Gottscho-
  Mark_203555232_gottscho_Lab2_2B.v
$ cp report.pdf
  Gottscho-Mark_203555232_gottscho_Lab2/Gottscho-Mark_203555232_gottscho
  _Lab2.pdf
$ chmod –R go+rx Gottscho-Mark_203555232_gottscho_Lab2/
$ tar –czf Gottscho-Mark_203555232_gottscho_Lab2_pin0072.tar.gz
  Gottscho-Mark_203555232_gottscho_Lab2/
$ chmod go+rx Gottscho-Mark_203555232_gottscho_Lab12_pin0072.tar.gz
$ cp Gottscho-Mark_203555232_gottscho_Lab2_pin0072.tar.gz
  /w/class.1/ee/ee201o/ee201ot2/submission/lab2/Gottscho-Mark_203555232_
  gottscho_Lab2_pin0072.tar.gz
```