Objective

Your objective in this lab is to develop a decoder module, which will convert a 4-bit binary value into a 7-bit control code for a 7-segment display. You will use your 8 instances decoder module to display values on the 8 HEX displays. For this project, you will simply display a counter that starts at zero and counts up. In later labs, you will use the same decoder modules you write now to display program output from your CPU.

Specific Tasks

1. Edit the `rtl/hexdecoder.sv` file to create your decoder module.
2. Edit the `rtl/simtop.sv` file to create a 32 bit counter which ticks up by one every clock cycle, and rests when `KEY[3]` is pressed.
3. Edit the `rtl/simtop.sv` file to instantiate 8 decoder modules.
4. Edit the `rtl/simtop.sv` file to connect each decoder's output to the corresponding HEX output from the `simtop` module.
5. Edit the `rtl/simtop.sv` file to connect each decoder's input to one 4 bit portion of the 32 bit counter.
6. Test your design by running `make gui`, be sure that the value on the HEX displays counts up when you advance the simulation time.
7. Edit `testbench/testtop.sv` to create a SystemVerilog based testbench for your HEX decoder.
8. Run your testbench in ModelSim with `make testbench`
9. When you are satisfied that your design works correctly, generate your submission file with `make pack`.Your submission file must be named `CSCE611_<Semester>_hex_<Your USC Username>.7z`. Your "USC username" is whatever you log into your university computer accounts with. For example, a student with the email address `jsmith@email.sc.edu` might turn in a file named `CSCE611_Fall2020_hex_jsmith.7z`. Please use the provided Make target, **do not pack your submission manually**.

Design Requirements

1. You must use the provided `simtop.sv` as your top-level module, and your design should execute with out error using `make gui`.
2. When running in the GUI simulator, advancing the simulation by one tick should increase the value shown on the HEX displays by 1. The value shown should be displayed in hexadecimal.
3. When `KEY 3` is pressed in the GUI simulator, the counter should reset to 0.

- **NOTE:** because of how the simulator simulates keys being pressed for random numbers of cycles, it may take a few dozen cycles before the `KEY[3]` line goes low again, it is OK if your design does not count during this period.

4. Your testbench should verify that each possible input value is correctly decoded. Note that you do not need to test your top-level module, you only need to test your hex decoder.

Rubric

- (A) 20 points – design correctly implements requirement #2.
- (B) 20 points – design correctly implements requirement #3.
- (C) 30 points – included test-bench correctly checks the behavior of the hex decoders.
- (D) 10 points – README describes how to run the test-bench.

Maximum score: 80 points.

Additionally, the following may cause you to lose points:

- Academic honesty violation, such as turning in another student's code.
- Code which does not compile.
- Failure to follow requirement #1.
- Attempts to subvert the automated testing system.

Hints

- This project is intended to be an introduction to the tools we will be using in this course. Make sure you understand it, since we'll be using the same methods for later assignments.
- You only need to instantiate one HEX decoder in your test bench.
- You can find the documentation for how the HEX displays are wired in the simulation handout.
- Remember the HEX displays are active-low.
- When using `$readmemh()` in your testbench, keep in mind that the path to the file to be loaded will be relative to the project root. So if your test vectors file is in `./testbench/vectors.dat`, you might do something like `$readmemh("./testbench/vectors.dat", vectors);`
- You don't have to use the template `hexdecoder.sv` file if you don't want to as long as your design meets the design requirements.

- You don't need to set up `config.sh` to find your memories as discussed in the simulator handout for this project, as you don't need to instantiate any memories.
- Because some aspects of the simulator that we will use for later projects (access to registers and program memory) are shorted out, you might see a number of "unused parameter" warnings while compiling. You can safely ignore these. For the same reason, the `make smoketest` target is disabled.

