EE 108
Fall 2020

Homework 1 Demo Questions

1. An FPGA is made up of a huge number of objects called “slices,” which consist of look-up tables (LUTs), flip-flops (we'll learn about these later), and some extra specialty hardware. The FPGA implements your design by setting the values of the LUTs and wiring the slices together, subject to complex constraints (that's why it takes so long to synthesize). After synthesizing, go to “Window -> Reports” and double click on “impl_1_place_report_utilization_0.”
    a. What fraction of the FPGA's slice LUTs did your implementation consume? You can find this information under section 2: Slice Logic Distribution.

    LUT as logic: used: 1325, Util% = 2.49%

    LUT as memory: used: 15 Util% = 0.09

    b. How many slices are occupied? The video and input wrapper code provided for you takes up most of this space, but it's still worth it to begin being aware of what kind of resources we're using.

    Slices occupied: 431

2. Open Window -> Reports -> synth_1_synth_synthesis_report_0 and text search (ctrl+f) for “latch”. This report will let us know if the design is accidentally inferring any latches with the warning “WARNING: [Synth ____] inferring latch for variable ‘_____’” where the blanks will change depending on what signal is being accidentally latched. If you have any of these warnings, you must fix all of them before submitting your files. If you did infer any latches, report where they were and how you removed them.

None found on first try!

3. On the left side panel, click on “Implementation -> Open Implemented Design.” After a bit of time, it will display a visualization of the FPGA itself, giving you an idea of what you actually built. You should see the slices being used highlighted in bright blue, and if you zoom in can even see the individual components being used in each slice. Submit a screenshot of the overall layout, and a screen shot of a single LUT slice.

EE 108
Fall 2020

4. Explain the difference between synthesizable and non-synthesizable Verilog. Give examples of pieces of syntax that you have used already that are non-synthesizable (you know of at least two).

Non-synthesizable Verilog is code that cannot be translated into hardware. The test benches are not synthesizable along with operations such as hash_out == pass_rom_out in the file verifier.v and in = in << 1 in encoder_tb.v. In the first case, we cannot encode "==" into hardware. In the second case, we cannot have the same variable on both sides of the equals sign.

5. Please include any comments about this lab so we can make it better in the future!

Make it clear that the rows in the table are the addresses for the users. Give an example for an if statement because in hash_round, figuring out that syntax was difficult and there was no clear description of the correct syntax.