ECE/CS 5710/6710 – Digital VLSI Design
Lab Assignment #7

Due Thursday Nov. 20th via canvas

1 Introduction

In this assignment you will, as a group, augment your library with various drive strengths of many of your cells. You will then use your library to synthesize a larger example. Finally you will learn how to place and route that example using the SOC Encounter place and route tool.

2 Background

Two aspects are fundamental to a good library. The proper functionality, as discussed in the previous lab, and providing cells with the proper drive strength. In this lab we will extend our base library to include multiple drive strengths for many of our cells.

The CAD tools must choose between cells of the same functionality but different drive strengths in order to optimize power and performance. Simply using the functionality of the cell is insufficient, as there can be multiple different transistor level designs that implement the same functionality. (Think of different transistor level XOR designs, and their various delays and properties.) Thus, all cells with identical functionality and similar design properties are placed into *classes*, called a **footprint**. The tools recognize that to improve the performance and power efficiency of a design, they can freely change the drive strength by picking a larger or smaller cell of the same *footprint*. As we extend our cell library with different drive strengths, we need to include information regarding the cell classifications called footprints. Note that the term footprint sounds like it is talking about the physical dimensions. This is not the case. In this application, cells with the same footprint have the same logical function, but may be of different physical size or transistor design.

3 Assignment

Following are the tasks for this laboratory:

1. Augment your library with the following cells. This means creating layout, cmos_sch, symbol, behavioral, extracted, analog_extracted, and abstract views, along with .lib,

.lef, and .v information. Add these cells to your Lib5710_xx library. Remember that all cells with the same function and design should have the same footprint. So, the INVXn cells should all have footprint "inv" and all non-inverting buffer cells should have the same footprint (i.e. buf), etc.

(a) INVX4 – an inverter with 4x drive

(b) INVX8 – an inverter with 8x drive

(c) BUFX4 – a non-inverting buffer with 4x drive. This is just a series of two inverters with the first being a 1x and the second being a 4x.

(d) BUFX8 – a non-inverting buffer with 8x drive. The first inverter should be a 2x and the second an 8x inverter.

(e) NANDX2 – a NAND gate with 2x drive

You will also eventually want to add other drive strengths for the other cells in your library. While it is not required for this lab, now is a convenient time to make these improvements to your library. You will also probably want to create a few widths of filler cells. If your original filler is named FILL1, consider adding a FILL4 and FILL8, that are 4 and 8 "standard cell widths" wide. (The standard cell width is 2.4μ.) This will permit the place and route tools to more efficiently fill open areas.

2. Update your library to add the .lib, .lef, and .v versions with the new cells you have defined.

3. Use your new Lib5710_xx library to synthesize the **controller** state machine from the MIPS example in your textbook. The controller.v behavioral file can be found in /uusoc/facility/cad_common/local/class/6710/F13/examples. Use a target speed of a 5ns clock period for your synthesis. Look at the report file to see if you were able to hit that target speed according to design compiler. Take the script file syn-script.tcl from the /uusoc/facility/cad_common/local/class/6710/F13/synopsys directory and suitably modify them for your own library and usage constraints. Do not use beh2str. Modify them properly to run through Design Compiler using syn-dc. You may also use the design compiler gui syn-dv (design vision) to drive the synthesis process through a GUI. Make sure to have .synopsys_dc.setup (copied from ../6710/F13/synopsys) in the directory from which you run syn-dc or syn-dv.

   As output from synthesis to the next step you'll need a structural Verilog file, and a .sdc delay constraints file from synthesis.

4. Place and route the structural Verilog file (controller_struct.v) using SOC Encounter EDI (cad-edi). Refer to Chapter 11 in Designing with Cadence and Synopsys for details on this process. Use your buffer cells (footprint buf) as your clock-tree synthesis cells. You should end up with a correctly placed and routed circuit with no geometry or connectivity errors, and with (at least) the following files as a result:

(a) **controller_soc.v** – the structural file from SOC Encounter EDI which includes the generated clock tree and the optimizations.

(b) **controller.def** – the design exchange format (DEF) file that describes the placed and routed circuit

(c) a report file for the final post-route optimization which shows whether the placed and routed circuit met the 5ns timing.

(d) **routed.enc** – The file which saves the SOC EDI state of the final placed and routed circuit.

5. Now read the controller.def file back into cad-ncsu to create the abstract view (Refer to Designing with Cadence and Synopsys Chapters 9 and 11 for details).

   (a) Make a new library named **control**. (Note: Be sure to attach the correct technology file!)

   (b) Import the **controller.def** file to this library as a layout view. Replace the abstract cell views with layout cell views and run DRC and Extract.

   (c) Import the **controller_soc.v** file (the structural Verilog file of the final placed and routed circuit from EDI) in as a schematic and symbol view.

   (d) Compare the schematic and extracted views with LVS and verify that they are the same.

4 Deliverables

Create a tar file containing all deliverables and turn in via canvas. Please only turn in a single tar file for each group.

1. A README file that describes the files in this directory.
2. A copy of your Lib5710_xx.lib, .lef, .db, and .v files.
3. An xv image of the results of your EDI encouter run showing the layout.
4. From soc/control, the structural verilog file from synthesis, the structural verilog file from SOC, the controller.def file from SOC, and the saved SOC file called routed.enc.
5. The LVS log showing that the layout and schematic match.
6. The timing report from Synopsys design compiler
7. The post-route timing report from SOC encounter.