ECE/CS 5710/6710 – Digital VLSI Design
Lab Assignment #6

Due Monday Nov 10th via canvas

1 Introduction

In this assignment you will, as a group, add to the library that you started with the last assignment. In particular you will add a D-type master-slave flip-flop, a filler cell, and at least five other cells of your choosing. The result is that your standard cell library will include at least 12 cells (including the five from Lab 5). Follow the procedure from the Lab 5 assignment to design and characterize your cells. This lab adds two new aspects to your cell library: a report from you on how you chose your additional cells, and a .lef file to go with your new expanded .lib file. The procedure for generating abstracts and .lef information are in Chapter 10 of *Design with Cadence and Synopsys*.

2 Background

Several researchers have claimed that a library with a small number of functional cells can perform almost as well as a cell library with a large inclusive set of functions. Next semester in 6770 you will be able to use such a library developed by Dr. Sechen at the University of Washington. His library contains few functions but *many* drive strengths. I am personally somewhat skeptical of the claims that such libraries can achieve similar performance and power efficiencies. I base this skepticism on my experience with Intel's world class library, and the difference I have observed by running some synthesis and simulation to compare between commercial and academic libraries. However, there is no doubt that the functionality that you pick for your library can have a huge affect on the performance and power efficiency of your designs, whether it is a small library with many drive strengths or a library with lots of functionality but fewer drive strengths. The library you pick can, in particularly, be highly optimized for your design. Since you will be forced based on time and resources to design a library with a small number of functions and drive strengths (not optimal in either axis), the choice of the functions you pick is critical. The cells you design should be based on the type of chip you will build. For instance, if you are building a design requiring addition or multiplication, the you will likely want to build half and full adder cells.

3 Details

You can try out different cell libraries without actually producing the physical design. This is done by creating .lib files that have your test cells.

As you probably figured out reading Chapter 8 on characterization, the .lib file is just an ASCII text file with some information in it about the cells. You can make up this information from thin air, and, after turning it into a .db file, synthesis tools will happily use those undesigned cells (assuming you generated the file with correct syntax). You could even use this technique to fiddle with the cells and see, for example, if Design Compiler would be more likely to use a cell if you could squeeze it a little smaller or make it a little faster.

In order to choose a good set of cells to add to your library, you should perform some experiments. Try out a few potential cells and see what results you get. Create four behavioral Verilog files and try them out with different cells available in the synthesis target. As an experiment, write behavioral Verilog for these four designs:

1. 2-input 4-bit-wide adder
2. 8:1 mux
3. 3:8 decoder
4. 4-bit counter with a 4-bit limit register. Counts from zero to the limit.

Establish a baseline by creating structural Verilog files (synthesizing with beh2str or with the syn-dc script) for the four behavioral Verilog targets. Find a figure of merit for the synthesis. (Area is a good one. Speed is also good, but depends on "reasonable" speed estimates of your undesigned and uncharacterized cells). You probably want to use the basic beh2str synthesis for this analysis just because it's quicker and easier to modify the target libraries because it requires fewer modifications to the scripts.

Now add a three input NOR and a three input NAND (these will also be fictitious) to the .lib file and see if the figure of merit improves.

Find out if design compiler likes non-inverting OR and AND gates.

Try and-or-invert (AOI) gates:

```
Y = ~((A & B) | (C & D);  // four input AOI22
Y = ~((A & B) | C);       // three input AOI21
```

Try or-and-invert (OAI) gates:

```
Y = ~((A | B) & (C | D);  // four input OAI22
Y = ~((A | B) & C);       // three input OAI21
```

Try two complex cells:

```
Y = A ^ B;                  // exclusive or
Y = (~C & A) | (C & B);     // 2:1 mux
```

A few years ago in this course all libraries were required to include an exclusive or (XOR) cell. Compare a library with AOI22 against one with an exclusive or. Modify the area (and other parameters) of these two fictitious cells to help determine if an XOR gate should be part of the library. Do the same comparison between the 2:1 mux and the AOI22.

There are two example libraries (in .lib and .db format) that you can use as starting points or templates for your exploration. They are both in /uusoc/facility/cad_common/local/class/6710/F13/synopsys/lib_files. The library named foo is a small example file from Chapter 8 of Design with Cadence and Synopsys. The library named osu05_stdcells is a slightly larger cell library from Oklahoma State University. You can use these as starting points as you fiddle with the cells that are available in the libraries. Or you can write your own new cell descriptions from scratch. Think about "reasonable" sizes and speeds for any new cells that you consider. Of course, you'll only be using your best guess at this point.

These steps are intended to help you to be as productive as possible! You should not initially physically design and layout out or characterize any cells before understanding their impact on your design! The process of experimenting with prospective cells in a library will help you create much better designs with less overall effort.

4 Assignment

4.1 Views

Follow the procedure from Lab 5 (Design with Cadence and Synopsys Chapters 6 and 8) to add at least seven more cells to your standard cell library. The cells should be designed in all views necessary to be included in the library. In particular:

1. **cmos_sch** – transistor level schematics with transistor sizes
2. **layout** – mask layout following cell template rules
3. **behavioral** – Verilog behavioral views for simulation
4. **symbol** – the schematic symbols
5. **extracted** and **analog_extracted** – from extraction and LVS
6. **.lib** – From ELC characterization

7. **verilog I/O** – Include the I/O description of your library's .v file. This is generated in step3 of the ELC flow, or done manually.

There are two new views that you will also need to make for this lab. These are created from the cad-abstract program:

1. **abstract** – This view is generated by the cad-abstract program and contains a graphical view of the bounding box, connection, and blockage information that the place and route tool uses.

2. **.lef** – This is a text file, also generated by cad-abstract, that is used by the place and route tool. This is similar to a .lib file but it has place and route information instead of synthesis information.

You should generate the cell macro descriptions using cad-abstract (Chapter 10), but use only one copy of the `TechHeader.lef` file that is in the /uusoc/facility/cad_common/local/Cadence/lef_files directory at the beginning of the file for technology information.

Note that you should generate these views with cad-abstract for *all* of the cells in your library, including the five from Lab 5. Therefore, you might want to wait until you have your new cells from this lab finished before running cad-abstract on all of them.

4.2 Cells

Experiment with Design Compiler and the results you achieve with different unimplmented cell functions as described in Section 3 of this assignment.

Following these experiments, add the following cells to your Lib5710-xx library:

1. A positive edge-triggered D-type master-slave flip-flop with an active-low asynchronous clear. See Lab 3 for details. You will probably need to redo the layout from Lab 3 to follow the standard cell template. You should include outputs for both Q and Qb. Inputs are D, CLK, CLRb.

2. A single-width filler cell. This cell is 2.4μ wide and is nothing but vdd, gnd, and well layers. It provides the place and router with a way to fill gaps in the standard cell rows. The cell boundary should be 2.4μ wide (one vertical routing grid wide), but the layers should overlap the official boundary by the amounts required in the template. This cell has no schematic view, and no simulation or lib view because it has no active devices in it. It does have layout, abstract, and .lef views.

3. Five more cells of your group's choice. You can decide what these cells are! In fact, how you choose must be documented as part of this lab. Think about what types of cells you might want to use if you were doing a design. Think about what special cells your project might need. Think about what type of cells might help the synthesis tool the most. Describe your synthesis experiments using the speculative .lef files from Section 3 in this lab.

5 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. List the cells that you created for your library.
2. Include documentation on the functionality of your cells: images of the schematic, layout, and functional simulations.
3. Final versions of your Lib5710_xx.lef, .lib, and .v files
4. Synopsys synthesis examples using your final 12-cell library on each of the four behavioral examples you used for deciding which cells to use. You can use the four examples in Section 3, or pick other examples that are more appropriate for your project. Turn in the Verilog code that you use as input to the synthesis process, and the structural Verilog code that results from synthesis for each of the four examples.
5. A report of your results. This report should list the cells you are including in your 12 cell library. This will be part of your final report at the end of the semester. This should describe:
   (a) the five basic cells you made in Lab 5
   (b) the flip-flop and filler cells
   (c) the five other cells that you chose. Justify the selection of these cells and any simulation that drove the selection.

   This report will also include information on your methodology and conclusions regarding which cells to include. Describe a simple model of how you think design compiler chooses between AOI22, the 2:1 mux, and exclusive or. Answer the questions:

   - Is exclusive or required?
   - When does design compiler prefer exclusive or?