

**f CS/EE 5710/6710 Digital VLSI**  
**CAD Assignment #5 (Group Assignment!)**  
**Due Friday October 5th, 5:00pm**

**Overview:** In this assignment you will, as a group, design and characterize a small (5-cell) standard cell library that has enough combinational cells to build combinational logic using Synopsys synthesis.

**Groups:** This is a group assignment. You should form groups as soon as possible. Groups can range from two to four members. Send email to teach-cs6710 with the names of the people in your group as soon as you know. If you're looking for a group, let me know and I'll try to match you up.

**Procedure:** Design five cells as described below following all the cell library template requirements in Section 5.7.1 of the Lab Manual. The distance between the center of the power and ground wires is fixed at 27 microns, for example, as defined in the template. The width of a cell is not fixed, but must be in increments of 2.4 microns. There are lots (!) of details in the Lab Manual.

For this lab you should make a new Cadence library called Lib6710\_xx, where xx is the number of your group, to hold your cells and only the cells. That is, Lib6710\_xx holds only your cells, and the designs that use those cells will be in a different library. This will make things easier later to keep your library separate from your project design. Once you tell me who is in your group, I'll tell you what your group number is. Also, once I know who is in the groups are I can set up UNIX groups for you so that you can use the group rw features to give group access to one single library. You will, of course, have to choose which group member has the library in their directory, but you should all be able to point to it in your Library Path. If you set your UNIX permissions to RWX for your group, and make sure that all the files in that directory are owned by the group, you should be all right.

The five cells you should add to your Lib6710\_xx library are:

- INVX1: standard 1X sized inverter
- NAND2X1: two-input NAND gate with standard size transistors (remember to take stacks into account when deciding on "standard" sizes)
- NOR2X1: two-input standard NOR gate

- TIEHI: A cell that provides a connection to vdd. There is only one output, and no inputs. The single output is a connection to vdd. This should be a resistive connection through a transistor so that the gate that it connects to is protected from direct connection to the power supply. The schematic is given at the end of this document.
- TIELO: A cell that provides a connection to gnd. See the schematic at the end of this document.

Your cells should have the following views: cmos\_sch, symbol, layout, behavioral (Verilog), extracted, and analog\_extracted.

Once you have the cells designed and simulated (and passing DRC and LVS), you should characterize the cells as described in Chapter 7 in the Lab Manual. This should result in a liberty format file for your five-cell library named Lib6710\_xx.lib. This is a very involved and somewhat picky procedure! I recommend using SignalStorm as described in Chapter 7, but this is a rather picky program in terms of getting the inputs just right. You can also simulate by hand or with scripts using Spectre directly if you have trouble using SignalStorm.

Once you have a Lib6710\_xx.lib, you should compile it into a Lib6710\_xx.db using Synopsys design\_compiler. This is described at the end of Chapter 7.

With a .db file of your five-cell library, you should demonstrate that you can synthesize a simple combinational circuit using Synopsys. You can use the simple beh2str script as described in the first section of Chapter 8, or you can use the more general dc\_shell script or design\_vision versions if you like (also described in Chapter 8). The idea for this assignment is just to make sure that your cell library is specified correctly so that synthesis will work!

**What to Turn In:** Use electronic handin for this assignment. Please do the following:

1 – Make a tar file of your Lib6710\_xx directory named Lib6710\_xx.tar and use handin to hand in that tar file. (one handin per group)

```
handin cs6710 CAD5 Lib6710_xx.tar
```

2 – Make a new directory called LibData\_xx (again, using your group number as xx). In that directory put the following

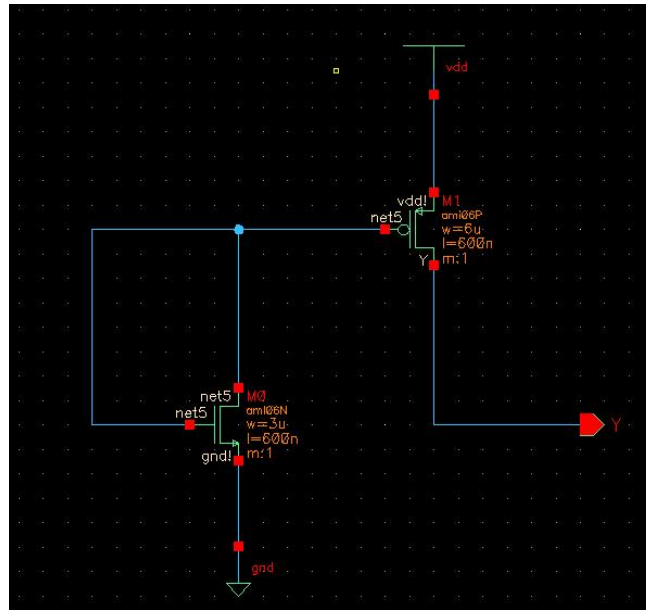
- a – The Verilog testbenches for each of the five cells
- b – PDF files of the waveforms of the Verilog simulations of each of your cells. Note that these simulations should be switch-level simulations of the cmos\_sch view, NOT simulations of the behavioral views. Make sure that your Verilog netlist settings are getting the correct views.
- c – A copy of your Lib6710\_xx.lib file that you generated using SignalStorm or Spectre simulation.
- d – The Verilog code that describes the combinational circuit that you synthesized. Note that because your library currently doesn't have a flip flop this must be a combinational circuit.
- e – The structural Verilog result from synthesis that demonstrates that the synthesis procedure used the cells in your library.
- f – A README file that describes what file is what.

Then make a tar file of that directory called LibData\_xx.tar and use handin to hand in that file too:

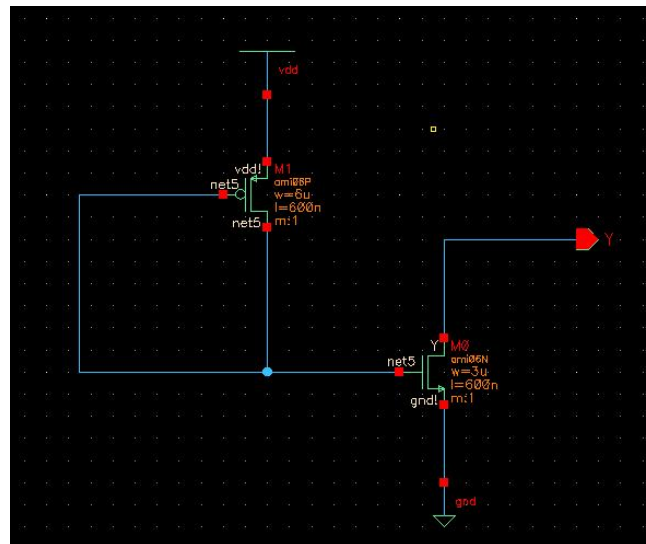
```
handin cs6710 CAD5 LibData_xx.tar
```

*Turn in only one set of documentation per group! The same person should use handin for both tar files so that they both go to the same handin directory.*

TIEHI schematic – diode-connected N-type provides a low voltage to the gate of the P-type that provides the pull-up output.



TIELO schematic – diode-connected P-type provides high voltage to gate of the N-type that provides the pull-down output.



There is an interesting issue when trying to simulate these cells with Verilog-XL: the switch level simulator doesn't know how to simulate diode-connected transistors. So, in order to correctly simulate these cells you need to trick it. Remember the trick we did in the register schematic of giving a node an attribute of trireg? Well, here we need to do the same thing, but we want the trireg node to be initialized to vdd or gnd instead of just holding the previous value. This is because from Verilog-XL's point of view, there is no

previous value! So, we need to give it a value to start with. A trireg that is initialized to Vdd is called **tri1** and a trireg that is initialized to gnd is called **tri0**. So, use the same technique that you used to add the netType attribute to the wire in CAD3 and give the internal nodes of the Tiehi and Tielo cells the right tri1 or tri0 attributes so that they will have the correct values in Verilog-XL. Note that these attributes have no effect on Spectre, so your analog simulations will correctly model the diode-connected transistor and everything will work out.

## SignalStorm Library Characterizer

Chapter 7 in the CAD manual goes in to lots of detail about what cell characterization is all about, what the format of the liberty (.lib) file is, and how to generate those files with SignalStorm Library Characterizer (SLC). Here's a very short cheat-sheet version of the process. You'll also need to read the CAD manual for the details...

1. Design all your cells. You'll need the following views for each cell:
  - a. **cmos\_sch** – transistor level schematics
  - b. **behavioral** – Verilog behavioral views
  - c. **symbol** – symbols for schematics
  - d. **layout** – mask layout according to the cell template
  - e. **extracted** – generated from the extract process and used for LVS
  - f. **analog\_extracted** – generated after successful LVS
2. Make a schematic that holds one copy of each of your five cells (**INVX1**, **NAND2X1**, **NOR2X1**, **TIEHI** and **TIELO**)
3. Use the Analog Environment to generate a Spectre netlist. Call it **foo.scs**.
4. Modify the **foo.scs** into **dut.scs** using a text editor or the **sp2slc** script in the /uusoc/facility/cad\_common/local/bin/F07 directory
5. Make an SLC directory in which to run **cad-slc**. Copy the **step1**, **step2**, **step3** files from /uusoc/facility/cad\_Common/local/class/6710/F07/cadence/SLC directory into your own SLC directory. Also put the **dut.scs** file in your SLC directory.
6. In your SLC directory, run **cad-slc** three times, once with each step file. Make sure that your scs file is named **dut.scs**, or change the step scripts to match the file name you used.
7. Finally use the **cad-alf2lib** script to convert the alf file from SLC to the .lib file you need. Make sure to properly edit your **footprints.def** file before running **cad-alf2lib**.