Elec 422: VLSI Design I

Homework 1

Fall 2000, September 12, 2000
Independent work is expected.
Due: 5pm, Monday, September 25, 2000.

For each of the following problems:

(a) Draw a logic diagram using Inverter, NAND, NOR, Transmission gate, etc. symbols,

(b) Draw a transistor-level circuit diagram indicating the NMOS and PMOS transistors,

(c) Draw a stick diagram utilizing alternate mirroring where possible,

(d) Use *magic* to draw the layout. Create a CIF file and use *cif2ps* or *pplot* to plot the layout.

(e) Create the circuit extractor files within *magic*. Complete the extraction of the circuit using *ext2sim -R -t !* for use with the *irsim* simulator,

(f) Perform an exhaustive *irsim* simulation, (as described below), testing all possible combinations of input and control signals. Please submit the *irsim* test vector files along with the graphical output of the *irsim* simulation. Use *irsim scmos100.prm file.sim* to invoke the scalable CMOS parameter file.

Notes:

- Assume that only the original signals are available. Any complement of a signal must be generated within the circuit with an inverter.
- All nodes within the circuit must be clearly labeled. Use *point* labels instead of box or line labels to label nodes in *magic*. Always label power as **Vdd!** and ground as **GND!** .
- Use compound gates, that is, and-or-invert structures where possible.
- Use static complementary logic only.
- Be sure to include substrate contacts in your stick diagram and magic layout.

Problem 1. Basic Logic Gate Design

Design a 2-input NAND and a 2-input NOR gate using $magic$. Use a style where Vdd and GND run horizontally in metal1, with input signals on the left and output signals on the right. Choose the same separation between Vdd and GND for both cells, approximately 50 to 60 lambda units.

Problem 2. Logic Expressions

Perform the above operations for the following logical expression:

$$g \;=\; \overline{(a \cdot b) + (c \cdot d \cdot e) + f}$$

Be sure to test all 64 possible input signal patterns.
(See the discussion in Weste, 2ed, pp. 15-17.)

Problem 3. Static Latch

Design a one-bit level-sensitive static latch. Label the input data as **data_in**, the control signal as **store**, and **storebar**, and the output as **data_out**. Use *magic* cell hierarchy to design the latch. In order to simplify the layout, use an inverter cell as a subcell in this problem. The top-level parent cell, **latch**, will contain two children cells, **inverter_0** and **inverter_1**. When using cell hierarchy in *magic*, make sure that the labels, **data_in**, **data_out**, **store**, and **storebar**, are placed on a square of paint contained in the top-level parent cell, **latch**.

Generate **storebar** from **store** by using a separate inverter. Place this inverter below the latch cell so that it can provide the **storebar** signal for the multi-bit latch in the next problem.

When testing the latch, the data and control signals *should not* both be changing at the same time. The data should remain stable or *valid* as the latch control signal goes low. Test the ability of the latch to store both "0" and "1".
(See the discussion in Weste, 2ed, pp. 19-21.)

Problem 4. Barrel Shifter

Design a 4-bit barrel shifter. Connect the output result lines to a 4-bit static latch. Use the latch cell designed in the above problem. A single inverter can be used to produce **storebar** from **store** to control the 4-bit latch. Although this is a complex circuit, please draw complete logic, transistor, and stick diagrams for all portions of the circuit.

In the *irsim* simulation, test the effectiveness of the latch to store data, and to provide isolation from changing data in the barrel shifter.

Note: Assume that only one of the barrel shifter control signals can be active at a given time. You do not need to enforce this in hardware; it is sufficient to enforce this condition in the test vector file. However, do provide an inverter to generate each of the **shiftbar** signals.

Also, you do not need to test all possible barrel shifter data combinations. However, a 4-bit barrel shifter will have a 7-bit input word. You should test all possible shifts (four cases), and the ability of the latch to store both a "0" and a "1" for each possible shift.
(See the discussion in Weste, 2ed, pp. 560-563.)