Lab 2: A Simple Decoder

Report Due Date: March 9 at 11:59 PM

Real-time Evaluation Date: March 6

- You will design a 2-to-4 decoder using only INV and NAND gates.
- You will test your decoder using MATLAB.

---

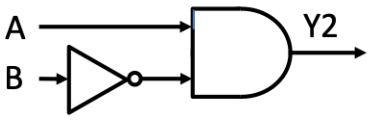
1 The Decoder Circuit

The decoder is a combinational circuit that takes a binary number with n bits as an input, and *decodes* it into a binary string with 2^n bits, where only one of the bits is active at a time (in this case, being active means being different from all the rest). The decoder is called "one-cold" if the active bit has a low (GND) value, while all the others have a high (V_{DD}) value. If the active bit has a high value while the rest stay low, then the decoder is said to be a "one-hot" decoder.

In this Lab 2, you will create a 2-to-4 one-hot decoder. This means that your circuit will have two inputs (A and B), and 4 outputs ($Y0$, $Y1$, $Y2$, and $Y3$), of which only one will be high at a time. The behavior of this circuit is detailed in the truth table on the right, where the entry of the "Active output" column corresponds to the only output from the decoder that will have a high (V_{DD}) value; all the rest will have a low (GND) value.

A	B	Active output
0	0	Y0
0	1	Y1
1	0	Y2
1	1	Y3

Implementing a 2-to4 decoder is fairly simple, and can be done by using only inverters and AND gates. E.g., the output $Y2$ can be generated using the circuit shown to the right: The output $Y2$ will only be high ($= 1$) in the case where $A = 1$ and $B = 0$. In all other cases, it will generate $Y2 = 0$. Note that CMOS is an inverting logic style, so to implement the AND gate shown to the right, you will have to use a NAND gate followed by an inverter. Similar circuits can be done to generate the outputs $Y0$,$Y1$, and $Y3$.



Your goal in this lab is to create a schematic *and* layout for a 2-to-4 decoder using only INV and NAND gates, while meeting delay and area constraints that we detail next.

2 Rules

1. Use $V_{\text{DD}} = 1.2\ V$ and standard operating conditions.
2. Adhere to the standard-cell dimensions we specified in Lab 1.
3. **Your decoder must use CMOS logic and only contain INV and NAND gates.**

4. **Each output of the decoder must be driving a 20 fF capacitive load.**
5. **The high-to-low propagation delay of all outputs must be not greater than 90 ps; the low-to-high propagation delay of all outputs must be not greater than 105 ps.**
6. **The area of your layout must be not greater than** $50\,\mu\mathbf{m}^2$**.**
7. Name your signals using the same uppercase names as in the decoder truth table.
8. All inputs to the decoder are generated from the provided verilog-A module (`lab2_signal_gen`) which runs at 2 GHz with 20 ps rise and fall times. You should unzip `lab2_signal_gen` in your ece4740 library and use it as an instance in your testbench schematic to generate the inputs of your decoder. You are not allowed to modify this module.
9. The 2-to-4 decoder layout must pass DRC and LVS.
10. You must test your design as detailed in the next section.

3 Testing

A major part of digital circuit design is testing. We now describe a way to test the functionality of your decoder through MATLAB. In particular, you will use the Cadence `cds_srr` function for MATLAB, which loads the Virtuoso simulation results into MATLAB. This data is then compared to a so-called *golden model* in MATLAB. If everything is working correctly, your circuit's output should match the golden model. To perform such comparison, follow the next steps:

1. Create your 2-to-4 decoder's testbench schematic.
2. Label the decoder's input and output nets. To do so, in the Schematic Editor window, go to "Create" → "Wire Name..." (or press "**L**" on your keyboard). As shown in Fig. 1, write the net's name in the "Names" field, and then click the desired net in the schematic window. ***Remark:*** *Make sure to use the names "A" and "B" (uppercase) for the decoder's inputs, as well as "Y0", "Y1", "Y2", and "Y3" (with uppercase "Y" and without quotes, obviously) for its outputs. The net names must be as described above so that they match the ones that have been hard-coded in the MATLAB testing file.*

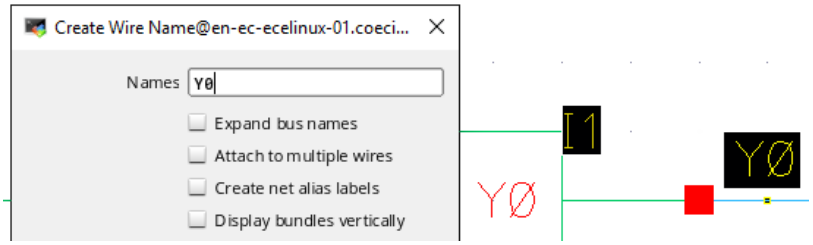


Figure 1: Adding a net label.

3. After adding the labels, verify that your schematic is similar to the one in Fig. 2.
4. Open the ADE simulator.
5. Set your simulation to be `tran` with a stop time of 10 ns.
6. *Make sure that your simulation project directory ("Setup">"Simulator/Directory/Host" in the ADE L window) is set to "*`~/Cadence/simulation`*"*. Do not forget to set the temperature to 25°C, and to use the `av_extracted` view for simulation. Furthermore, if you want to see the plots of the input and output signals of your decoder, remember to add the corresponding nets to the ADE outputs.

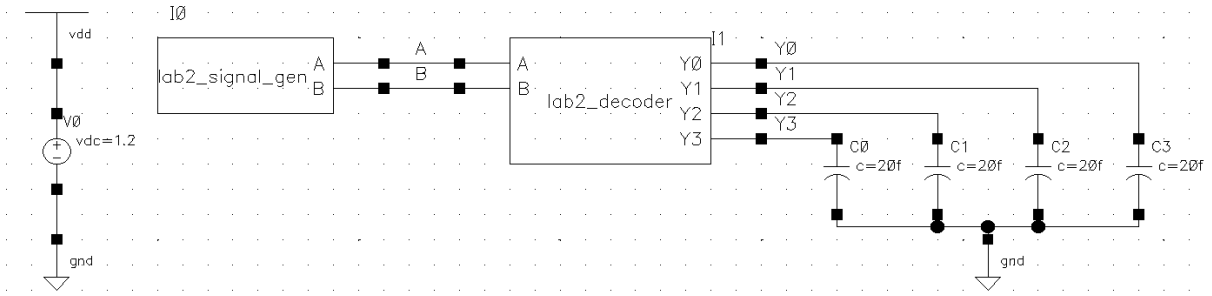


Figure 2: Decoder testbench with net labels.

7. Run your simulation.
8. Download the "lab2_testing.m" script. You can do so using Filezilla or MobaXTerm (if connecting remotely to the `ecelinux` servers), or by directly downloading the file to the computer (if you are working in Phillips 314). Save it to a location that you will remember later.
9. Start MATLAB by typing in the terminal: `matlab &`
10. Using MATLAB's navigation bar (left side of the MATLAB window), go to the directory where you saved the "lab2_testing.m" file and open it.
11. Modify line 8 of the MATLAB script so the variable `tb_name` contains the name of your testbench cell in Cadence Virtuoso.
12. Run the MATLAB script by pressing the F5 key or the green play button on the top bar.

    This script tests the functionality of your decoder circuit. To do so, the script reads your decoder's inputs and outputs directly from Cadence, samples the decoder's output values 105 ps after the inputs crossed 0.6 V, and determines if the output matches with the expected response from the MATLAB golden model. MATLAB is able to access the results from the Cadence Virtuoso simulation using the `cds_srr` function; please have a detailed look at the script and use the MATLAB `help` command to better understand how this function works. ***Important: You will have to test all future designs in this course using the same approach.***

Optional: Exporting your data to a CSV file

In the previous section, we used the MATLAB function from Cadence, `cds_srr`. However, what happens if you do not have access to MATLAB, but you still want to access the simulation data from another programming language? One way of doing this is by exporting the simulation data from Cadence Virtuoso into a comma-separated-values (CSV) file. To create the CSV file, you need to go to the plot generated by the ADE simulator. Then, on the left side of the window, select all the signals that you would like to export, right-click and press: "Send To"→"Table"→"New Window". After doing this, a new window will open. In this new window, press "File" → "Export" to save the table as a CSV file.

4 Grading

Area (10 pts): The area is calculated by the smallest rectangular bounding box (from half of the upper-most power rail to half of the lower-most power rail) in μm^2. The area of your 2-to-4 decoder must be not greater

than 50 μm^2.

Delay (10 pts): The high-to-low propagation delay of all your outputs must be not greater than 90 ps; the low-to-high propagation delay of all outputs must be not greater than 105 ps.

Testing (10 pts): Your decoder circuit must be fully functional and tested for all inputs.

Layout (10 pts): We will verify the quality of your layout (e.g., that standard-cell design rules were properly used, that adjacent metal layers are routed in orthogonal directions, that nearby wires do not run in parallel for long distances, that a proper amount of metal layers is used, etc.).

Real-time Evaluation (30 pts): An evaluation time will be scheduled. First, the Prof./TA will verify the delay, area, energy, DRC, and LVS. They will also ask you questions about your design choices or conceptual questions (e.g., how do the load capacitors, supply voltage, or transistor sizes affect your benchmark metric). You will receive an individual grade in the range 0-to-5 based on this evaluation. **Evaluations will take place March 6**. Groups will sign-up for evaluation time slots on Canvas as in Lab 1.

Report (25 pts): Fill out the PowerPoint report template posted on Canvas. Make sure your schematics and plots are clear and legible. You can label Cadence screenshots or draw the schematics in a program of your choice. Clearly label your screenshots (layout and schematic).

Follow the Rules (5 pts): Turn in a *single* zip file with your schematics, symbols, layouts, MATLAB testbench, and the report. Convert the report to pdf (but include the original PowerPoint file too). Name the zip file **Lab2-(NetID).zip** (without parentheses). All files, including the report, are due at 11:59 PM on March 9.

Important: We will try all the submitted designs, compare it to the reported numbers, and test the functionality. In case the design does not pass DRC, LVS, or testing, and the schematics, symbols, and the layout are not submitted, you will get penalized by 20%. If you decide to cheat with benchmark numbers, fake LVS/DRC pass reports, etc., you will receive zero points for this lab.

Last updated: February 18, 2020 by Thomas Tapen / Oscar Castañeda / Alexandra Gallyas / CS
Special thanks to Olalekan Afuye for discovering the `cds_srr` function!