Lab 2: A simple decoder

Report Due Date: March 12 at 6:00 pm

- Your goal is to design a 3:8 decoder having *minimum area* using only INV, NAND, and NOR gates.
- You have to test whether your design produces the correct outputs.
- The worst-case propagation delay of your design must not exceed 52 ps.
- Every decoder output must drive a 20 fF load capacitance.

---

1 Decoder Truth Table

A	B	C	Decode (output)
0	0	0	Y0
0	0	1	Y1
0	1	0	Y2
0	1	1	Y3
1	0	0	Y4
1	0	1	Y5
1	1	0	Y6
1	1	1	Y7

Explanation: Only the listed output should be high (VDD), the other outputs should be low (GND). In other words, the circuit to be designed has 8 outputs, where only one of the outputs is high while the others are low. The outputs must have full rail-to-rail swing (no voltage drops allowed) and only static CMOS is acceptable.

1. Name your signals using the same uppercase names as the Decoder Truth Table
2. You must layout (pass DRC & LVS) at least two of the following logic gates even if you only use one type in your final design: NAND2, NOR2, NAND3, NOR3, INV
3. All inputs to the decoder are generated from the provided verilog-A module (`clock_gen`) which runs at 5 GHz with 20 ps rise and fall times. You should unzip `clock_gen` in your ece4740 library and use it as an instance in your testbench schematic to generate the inputs of your decoder
4. Assume that all inputs have the input capacitance of a minimum-sized inverter. Use this information for the purpose of sizing calculations only. *Hint: Use your Lab 1 report*

5. Power rails must be at least 400 nm wide
6. VDD = 1.2 V
7. The decoder layout must pass DRC and LVS
8. You must calculate the rise and fall delays of each output signal
9. You must test your design as detailed below

2 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 signals generated during a Virtuoso simulation into MATLAB. This data is then compared to a so-called *golden model* in MATLAB. The hope is that your circuit's output match the golden model.

After you complete your decoder testbench schematic, label the decoder's input and output nets using "Create" → "Wire Name..." (or by pressing "L"). As shown in Fig. 1, write the net's name in the "Names" field, and then click the desired net in the schematic window. *Make sure to use the names "A", "B", and "C" (uppercase) for the decoder's inputs, as well as "Y0", "Y1", . . ., and "Y7" (with uppercase "Y") 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. After finishing this step, your schematic should be similar to the one in Fig. 2.

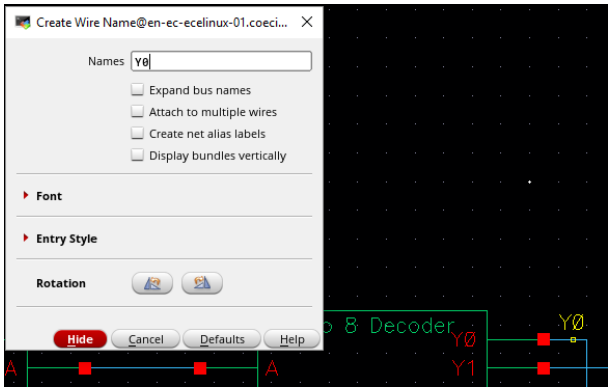


Figure 1: Adding a net label in Cadence Virtuoso.

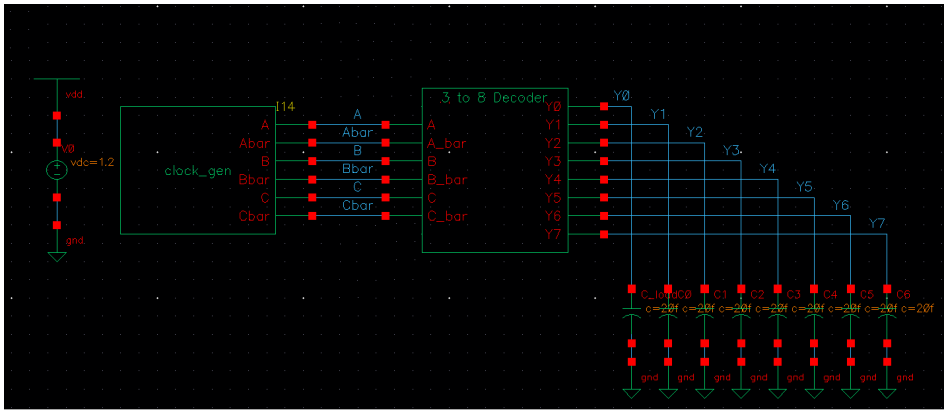


Figure 2: Simulation testbench with net labels.

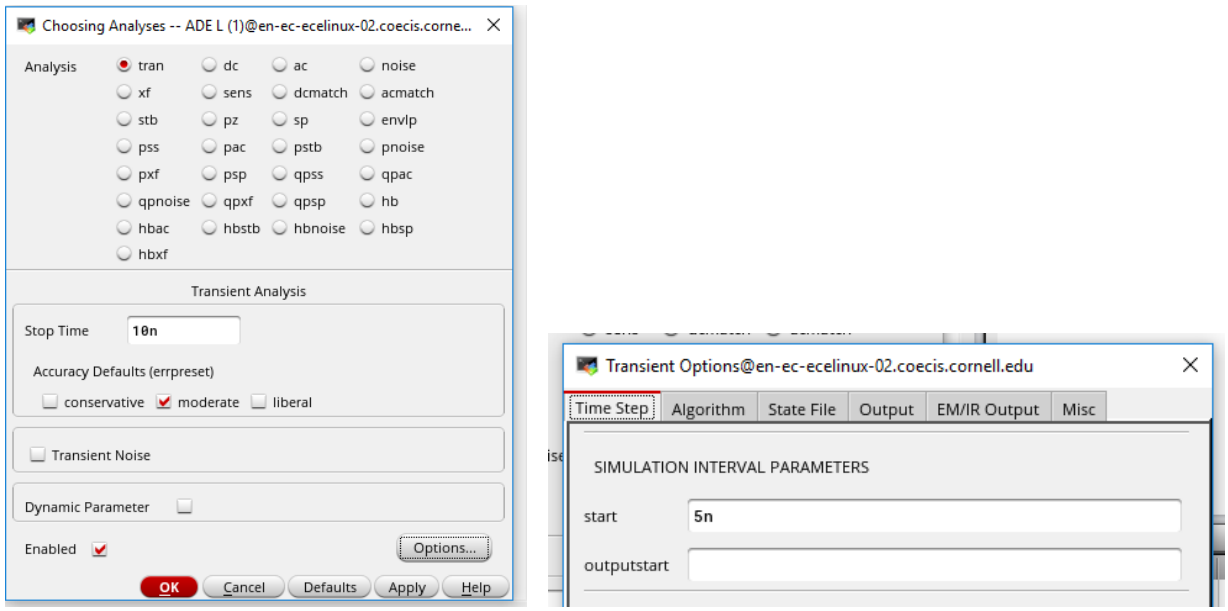


Figure 3: ADE Analyses settings

Next, open the ADE simulator. Since there is a 5 ns delay for the input `clock_gen`, we will be sampling the output signals from 5 ns to 10 ns (as opposed to 0 ns to 10 ns). Set your simulation to be `tran` with a stop time of 10 ns. In addition, to make sure Cadence starts sampling at 5 ns, press "Options..." in the "Choosing Analyses", and in "Time Step" tab, put "5n" in the start tab as shown in Fig. 3. *Make sure that your simulation project directory ("Setup">"Simulator/Directory/Host" in the ADE L window) is set to "~/simulation".* Do not forget to set the temperature to 25°C. 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. Once you are all set, press OK and run your ADE simulation.

The next step is to open MATLAB. Before you do so, you need to download the "lab2_testing.m" script to your computer: You can do so using Filezilla or MobaXTerm (if connecting remotely to the `ecelinux` servers) or directly downloading the file to your computer (if you are working in Phillips 314). Save it to a location that you will remember later. You also need to open MATLAB on the same computer that runs Cadence Virtuoso. To setup the path to MATLAB, enter the following command in the terminal:

```
echo "export PATH=$PATH:/opt/matlab/R2016A/bin/" >> ${HOME}/.bashrc
```

Then start MATLAB by typing in the terminal: `matlab &`

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. Modify line 8 of the script so the variable `tb_name` contains the name of your testbench cell in Cadence Virtuoso. Then, 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 input/output directly from Cadence, samples the midpoint of the decoder's output values, and determines if it matches with the expected response from the MATLAB golden model. MATLAB is able to access the results from the Cadence Virtuoso simulation thanks to the `cds_srr` function; please have a detailed look at the script and use the MATLAB `help` command to better understand this function. You will have to test all future designs in this course using the same approach.

3 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, select all the signals that you would like to export, right-click and press: "Send To", "Table", and "New Window". See Fig. 4 for reference. After doing this, you will obtain a new window like the one in Fig. 5. Then press "File" → "Export" to save the table in this window as a CSV file.

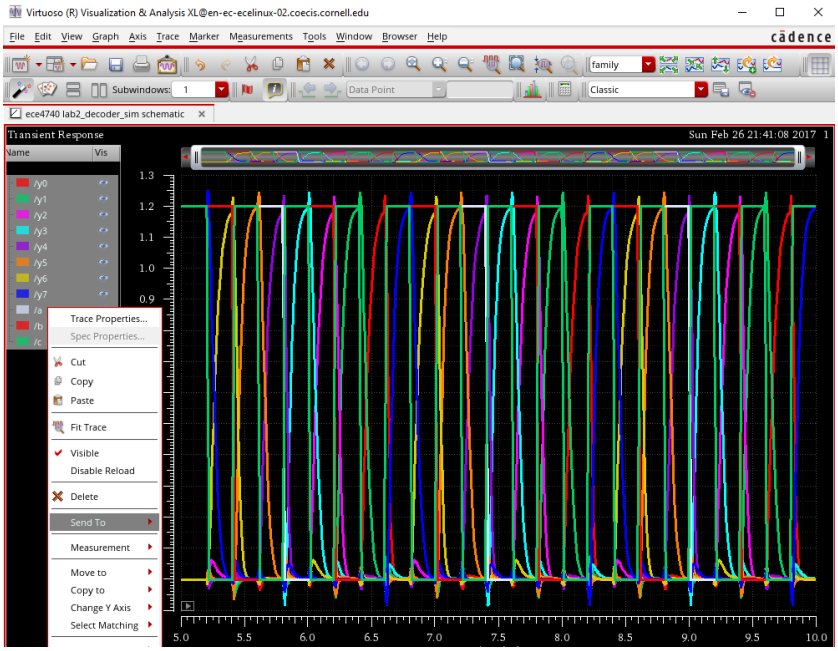
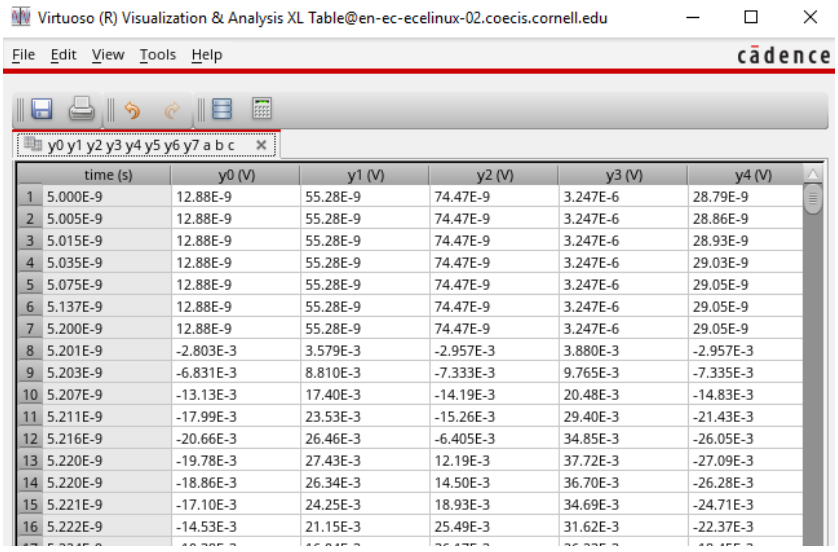


Figure 4: ADE simulation result.



	time (s)	y0 (V)	y1 (V)	y2 (V)	y3 (V)	y4 (V)
1	5.000E-9	12.88E-9	55.28E-9	74.47E-9	3.247E-6	28.79E-9
2	5.005E-9	12.88E-9	55.28E-9	74.47E-9	3.247E-6	28.86E-9
3	5.015E-9	12.88E-9	55.28E-9	74.47E-9	3.247E-6	28.93E-9
4	5.035E-9	12.88E-9	55.28E-9	74.47E-9	3.247E-6	29.03E-9
5	5.075E-9	12.88E-9	55.28E-9	74.47E-9	3.247E-6	29.05E-9
6	5.137E-9	12.88E-9	55.28E-9	74.47E-9	3.247E-6	29.05E-9
7	5.200E-9	12.88E-9	55.28E-9	74.47E-9	3.247E-6	29.05E-9
8	5.201E-9	-2.803E-3	3.579E-3	-2.957E-3	3.880E-3	-2.957E-3
9	5.203E-9	-6.831E-3	8.810E-3	-7.333E-3	9.765E-3	-7.335E-3
10	5.207E-9	-13.13E-3	17.40E-3	-14.19E-3	20.48E-3	-14.83E-3
11	5.211E-9	-17.99E-3	23.53E-3	-15.26E-3	29.40E-3	-21.43E-3
12	5.216E-9	-20.66E-3	26.46E-3	-6.405E-3	34.85E-3	-26.05E-3
13	5.220E-9	-19.78E-3	27.43E-3	12.19E-3	37.72E-3	-27.09E-3
14	5.220E-9	-18.86E-3	26.34E-3	14.50E-3	36.70E-3	-26.28E-3
15	5.221E-9	-17.10E-3	24.25E-3	18.93E-3	34.69E-3	-24.71E-3
16	5.222E-9	-14.53E-3	21.15E-3	25.49E-3	31.62E-3	-22.37E-3

Figure 5: ADE Table view.

4 Grading

Area: The area is calculated by the smallest rectangular bounding box in μm^2.
Delay: 50% worst-case delay calculated from the verilog-A output to the decoder output.
Energy: The integral of supply current from 5.2 ns to 6.8 ns times the supply voltage.
Testing: Your decoder circuit must be fully functional and tested for all inputs.

Grading: 5 points for real-time evaluation, 5 points for layout, 5 points for area, 5 points for meeting the delay constraint (0 points for not meeting it), 5 points for energy, and 5 points for testing (0 points for a design that does not match the golden model). The score distributions are given by:

Score	5	4	3	2	1	0
Area	0–79μm^2	80–149μm^2	150–299μm^2	300–499μm^2	500μm^2–1m^2	no layout
Energy	0–399fJ	400fJ–799fJ	800fJ–2pJ	2–4pJ	4–8pJ	no result

Hints:

- The internet is your friend: there are multiple solutions to implement the decoder logic. Some solutions simplify design and layout, and also make it smaller, faster, and more energy efficient.
- You can freely set your PMOS:NMOS ratio.
- Calculate the rising and falling delays for every transition generated by `clock_gen`.

Grading consists of a real-time evaluation by the prof or a TA, as well as the report.

Real-time Evaluation: 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 the evaluation.

Report: Fill out the PowerPoint report template posted on blackboard. 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).

Turn in a *single* zip file with your schematics, symbols, layouts, and the report. Name the zip file Lab2-(group number). The report is due at 6:00 pm on March 12.

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, the entire group gets penalized by 20%. If you decide to cheat with benchmark numbers, the entire group will receive zero points for this lab.

Last updated: February 26, 2018 by Oscar Castañeda/cj/cs
Special thanks to Olalekan Afuye for discovering the `cds_srr` function!