ECE 2544: Fundamentals of Digital Systems
Learning Experience A (Minor): Transistor-Level Modelling of Logic Gates and Logic Circuits

Read the entire specification before you begin working on this project!

0. Preliminaries

Honor Code Requirements

You must comply with all provisions of Virginia Tech's Honor System. Your Verilog code and submitted documentation must be your own work. Your design, coding, debugging, and testing must all derive from your own effort. You may ask other students general questions about how ModelSim Intel FPGA Starter Edition works, but you **are not** allowed to ask questions about your design or implementation of **anyone** except for your instructor or a 2544 GTA or ULA. It is an Honor Code Violation to share **any element of your design or implementation** with **any other person**, or to copy **any element of your work** from **any other person's design** – either from paper, from a computer file, or from the Internet.

Ask your instructor if you have any questions about what is and is not allowed under the Honor Code.

Objectives

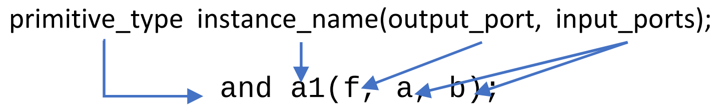
The purpose of this assignment is to model logic gates and logic circuits at the level of CMOS transistors. After completing this assignment, you will have gained familiarity with writing modules using CMOS transistors in the Verilog HDL, and with general principles that will transfer to writing other structural models using primitives.

Preparation

You must have access to a computer that can run ModelSim Intel FPGA Starter Edition. You should refer to the 2544 Lab Manual for instructions on using ModelSim. This assignment *does not* use the DE-10 Lite board or Quartus. Consult your class notes and the textbook to review CMOS formulation of logic gates.

Background

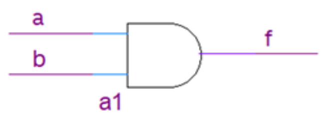
Recall that the Verilog HDL allows the designer to describe gate-level primitives. When modeling a primitive gate in Verilog, the order in which ports are placed on the gate's port list matters:



primitive_type instance_name(output_port, input_ports);

```
and a1(f, a, b);
```

(a)



(b)

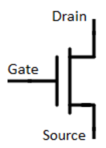
Figure 1: (a) Typical Verilog syntax for a primitive gate; (b) The corresponding gate, as a schematic

The Verilog HDL likewise contains primitives that describe the CMOS transistors that we have studied in class:

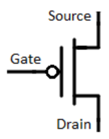
```
nmos instance_name(drain, source, gate);

pmos instance_name(drain, source, gate);
```

(a)



(b)



(c)

Figure 2: (a) Verilog syntax for the NMOS and PMOS transistors; (b) The corresponding NMOS transistor; (c) The corresponding PMOS transistor

As we saw with gate primitives, we can use Verilog `wires` to create structural models of interconnected NMOS and PMOS transistors. When connected correctly to each other and to the proper supply voltages, these interconnected transistor networks will function as CMOS logic gates.

Keep in mind, ModelSim is a simulation tool. It therefore idealizes certain gate and circuit elements. As you progress through the ECE curriculum and your career, you will use a wide range of simulation tools; for each, you should consider whether you are modeling behavior that corresponds to real-world behavior.

Transistors work by having the particular "junctions" (such as the one between the gate and the source) "overcome" a particular threshold voltage. The level of the threshold voltage depends, among other things, on the semiconductor material that is used to make the gate. For NMOS transistors, we generally have to have a voltage drop across the gate-source junction that is greater than some value V_T. As you might expect, we generally have to have a voltage drop across the gate-source junction that is less than some value $-V_T$ in a PMOS transistor.

In ModelSim, the NMOS and PMOS transistors are modeled with threshold voltages that equal 0, which does not exactly match reality. This quirk allows you to simulate some circuits in ModelSim that don't correspond to what you can do with actual NMOS and PMOS transistors in the real world. (For example, assuming the source of an NMOS transistor is connected to 5V, or assuming the source of a PMOS transistor is connected to 0V.)

For this Learning Experience, you must follow the principles for modeling CMOS logic as it was presented to you in class. This will produce circuits that work in ModelSim, and would also work in real gates. If you don't follow those principles, your logic might provide correct simulation results, but will not correspond to real-world behavior, and will not receive credit.

1. Assignment Instructions

The following steps will guide you through the creation of Verilog modules for the primitive logic gates. ***As you will find throughout this course, following the directions carefully, and in order, will be essential to successfully and efficiently completing the Learning Experiences.***

Step 1
Locate the Verilog source file named **TransistorModels_YOURPID.v**. This file contains the module declarations for the CMOS gates that you will be implementing. *Rename the file, replacing YOURPID with your Virginia Tech PID.* Your Virginia Tech PID is the part of your student e-mail address that comes before *@vt.edu*. Do not use your 9-digit student number. For example, if your e-mail address is johnsmith@vt.edu, then you would save your file as **TransistorModels_johnsmith.v**.

> ***WARNING****: ModelSim has a tendency to save files without extensions, even when you choose "Save As" and select a file type. When you save files in ModelSim, explicitly add the extension .v to the names of your Verilog source and test bench files in the File Name dialog box before you hit Save.*

Open a new project in ModelSim. Add your renamed transistor models file and the file **tb_TransistorModels.v** to the project. The file **tb_TransistorModels.v** is called a *test bench*; it is a model for a circuit written in Verilog that, when simulated, tests the behavior of other circuit models that have been instantiated within it.

> ***WARNING****: When you create folders and projects in ModelSim, keep in mind that the names of files and folders should* ***not*** *include spaces.*
>
> ***WARNING****: Normally you can open a Verilog source file in ModelSim by clicking on its name in the Project window. Sometimes doing this will cause the file to open in a text editor. When this happens, you can "anchor" a file in ModelSim by choosing* ***File > Open****, and then choosing any file in the active Project. Once one file is opened, others will open in ModelSim by clicking on their names in the Project window.*

When you first open the file that will contain your transistor models in ModelSim, remember to complete the included header block so that it contains the appropriate information. Follow the example below:

```
// Module:       adder1bit
// Author:       J.S. Thweatt
// Date Created: 29 July 2019
// Version:      1 (Date Last Modified: 29 July 2019)
// Description:  This file contains a 1-bit full adder implemented
//               structurally using built-in primitive operators.
```

Figure 3: Example module header - Use module headers as a minimum indication of the contents of the model you are making.

***WARNING**: For the steps that remain, **YOU MUST ENSURE THAT THE MODULES ARE NAMED AS INDICATED IN EACH STEP, SO THAT YOUR VERILOG FILE CAN BE VALIDATED**. Failure to name your modules as required may result in receiving no credit for your work.*

Step 2

In your renamed transistor models Verilog file – the file originally named **TransistorModels_YOURPID.v** – complete the modules for each of the gates below *using only CMOS transistors – that is, only the* `nmos` *and* `pmos` *transistor primitives described in the Preliminaries section of this assignment*.

- A NOT gate; use the module named **not_t**.
- A 2-input NAND gate; use the module named **nand_t**.
- A 2-input NOR gate; use the module named **nor_t**.

***Tip:** Recall that the NOT, NAND, and NOR gates have structures that derive from the principles behind NMOS and PMOS transistors. Review the principles concerning what it means for particular transistors types to be on or off, and what it means for particular transistor types to be in parallel or in series.*

Step 3

In your renamed transistor models Verilog file, complete the modules for each of the gates below *using only CMOS transistors*. Again, *only the* `nmos` *and* `pmos` *transistor primitives described in the Preliminaries section of this assignment* may be used.

- A 2-input AND gate; use the module named **and_t**.
- A 2-input OR gate; use the module named **or_t**.

***Tip:** In turn, recall that the AND and OR gates have structures that derive from the NOT, NAND, and NOR gate structures.*

Step 4

In your renamed transistor models Verilog file, complete the modules for a 2-input XNOR gate *using only CMOS transistors*. Use the module named **xnor_t**.

***Tip:** There are multiple ways to define an XNOR gate as an interconnection of other gates. You should keep transistor efficiency in mind when designing and modeling your solution.*

<u>Step 5</u>
Using Boolean algebra or other simplification techniques, the Boolean expression

$$F(a, b, c) = a'b'c' + ab'c + a'bc + a'b'c$$

simplifies as follows:

$$F(a, b, c) = a'b' + a'c + b'c$$

(You need not demonstrate this as part of this assignment.)

In your renamed transistor models Verilog file, complete the module for a circuit *using only CMOS transistors* that models the *simplified version* of the Boolean expression shown above. Use the module named **sop_t**.

<u>Step 6</u>
Compile your project, following the instructions given in the ECE 2544 Lab Manual. If the compiler gives errors or warnings, resolve them now before you proceed. The compiler should return 0 warnings and 0 errors.

***Tip:** In most cases, for each compile that fails, you can double click on the line in the **Transcript** panel to bring up a window showing the reasons for the unsuccessful compile. Consult the Lab Manual for a picture illustrating this.*

After you have debugged for syntax errors, simulate the *test bench*. **Do not simulate the individual modules**. The test bench provides stimulus values that apply inputs to your modules, and "gathers" together the outputs so that you may view them as a waveform.

***Tip:** The test bench contains instances of the modules you wrote in the Verilog file containing your transistor models. As this course develops, you will write modules and then instantiate them in higher-level modules as part of a design hierarchy.*

*While there are similarities between function calls and instances, **the test bench is not calling the modules, as happens in high-level language code**. Instead, each instance represents a copy of the module in question, wired inside the test bench and using the ports of the test bench as its inputs and outputs.*

***Tip:** Pressing the **Run** button in the **Wave** window only causes the simulation to advance by the time interval in the small window* 100 ps *to the left of the **Run** button.*

*Pressing the **Run** button several times will eventually cause the simulation to advance through all of the test values in the simulation. However, this isn't a particularly effective way to generate the simulation result for this test bench.*

*Instead, you should press the **Run -All** button to advance the simulation through the entire range of time described by the test bench.*

Use the simulated results to determine that each of your circuits behaves as expected. Follow the instructions in the Lab Manual to generate a waveform of the test bench. There you will see signals and outputs of the modules written in the solution file. *If you used the module names contained in this specification – names that are already contained in the Verilog file containing your transistor models – you won't need to alter the test bench.*

***Tip:** The waveform should be entirely green. A red waveform indicates an unknown value. A blue waveform indicates an unassigned, or high impedance state. You will learn more about high impedance states later in the semester, but if you come across a blue waveform, it indicates that none of the transistors connected to the output are on, and the output is not set to either logic-1 or to logic-0.*

Tip: *A Verilog source file that compiles free of syntax errors is no guarantee that your models are free of structural or logical errors. Debug your circuits until each one produces the output values that you would expect a gate or circuit of the sort described in this specification should have.*

Step 7

Once your waveform displays correct behavior, screenshot the waveform and save it as **LE_A_waveform_YOURPID.PNG**. As you did with your Verilog source file, replace YOURPID with your Virginia Tech PID. You may save the screenshot of the waveform to PDF or as a JPEG if you wish.

Tip: *Before you make the screen capture of your waveform, there are many useful things that you can do to improve the clarity of your waveform:*

- *Choose **Format > Toggle Leaf Names** to shorten the field names of the input and output ports as they appear in the waveform window.*
- *Right-click on a port name and choose **Add > New Divider**. This allows you to section your signals and make them easy to follow. For example, you might add dividers to indicate which ports are gate inputs and which ones are gate outputs. You can edit a divider by double-clicking on it. You can delete a divider by highlighting it and hitting the Delete key.*
- *You can highlight one or more columns, right-click on one, and choose **Properties**. On the **Format** tab, you can change the height of your rows to better space them out.*
- *You can drag the borders of the windows containing the port names and signal values to make them wider or narrower. It's a good idea to devote as much space to the actual waveforms as possible, and making those windows narrower can help with that. Make sure that the full names of your ports and the values of your signals are still easy to read once you resize those windows.*
- *You can change the order of the ports by clicking a port name to highlight it and then dragging it to the place where you want it to appear. ModelSim doesn't always list your ports in the order that you might like, so you can do this to change how your waveform is organized.*
- *You can Zoom In or out Zoom Out on a waveform, either to focus on a particular part of it, or to place the whole waveform into the area you want screen capture. It's also useful to Zoom Full – this makes a best fit of the waveform's timescale to the visible area of the waveform window.*
- *As an alternative to making screen captures, choose **Print** and then select whichever option on your computer allows you to print to PDF in the **Name** field. On my computer, the option is "Microsoft Print to PDF" but on your computer it might be different.*

2. Submission

Prior to the deadline indicated by your instructor, submit the following files to Canvas:

- **TransistorModels_YOURPID.v**
- **LE_A_waveform_YOURPID.PNG** (or .PDF or .JPEG)