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 Questa 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 the 2544 course support team. 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 transistor level using complementary MOS (CMOS) processes. 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. Knowledge of transistor usage in logic gates will be required for the remainder of this course.

Preparation

You must have access to a computer that can run Questa Intel FPGA *Starter* Edition. You should refer to the 2544 Lab Manual for instructions on using Questa. 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. CMOS technology is based on complementary pairs of n-type (NMOS) and p-type (PMOS) MOSFETs. All logic gates in this assignment and this course should be implemented in CMOS.

> ***REPEAT**: This assignment does NOT use the DE-10 Lite board or Quartus. To open Questa, search for "Questa" from your start menu. Be sure to use the section of the lab manual that* Creating and Simulating Verilog Source Files in Questa. *Again, **not Quartus**.*
>
> ***REPEAT**: All circuits in this assignment should be implemented using complementary pairs of n-type (NMOS) and p-type (PMOS) MOSFETs.*

Background

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:

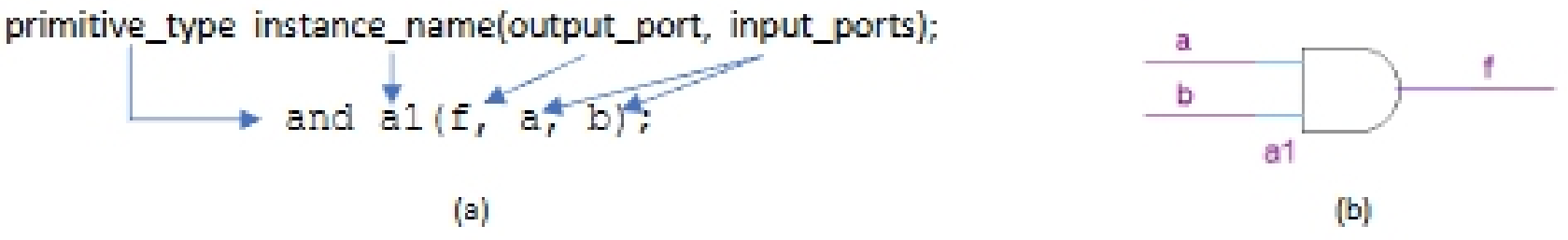


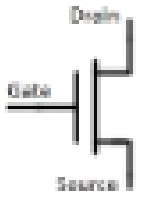
Figure 1: (a) Typical Verilog syntax for a primitive gate; (b) The corresponding gate, as a schematic

The Verilog HDL likewise contains primitives that describe the CMOS transistors that we have studied in class. Just as with gates, the order in which ports are placed on the transistor's port list matters.

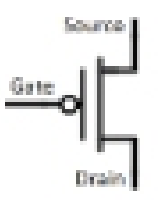
```
nmos instance_name(drain, source, gate);

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

(a)

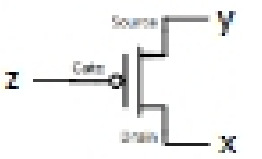


(b)



(c)

```
pmos P1(x, y, z);
```



(d)

Figure 2: (a) Verilog syntax for the NMOS and PMOS transistors; (b) The corresponding NMOS transistor; (c) The corresponding PMOS transistor; (d) Illustration of the effect of port list order on circuit connections.

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, Quartus 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, in a PMOS transistor, we generally have to have a voltage drop across the gate-source junction that is less than some value $-V_T$.

> ***WARNING**: In Questa, 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 Questa 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 Questa, 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

Create a local working directory (folder) for this project that does not contain spaces in the pathname, e.g. c:\ece2544\le_a. Save the two Verilog files provided with the assignment (TransistorModels_YOURPID.v and tb_TransistorModels.v) in the working folder.

The file **TransistorModels_YOURPID.v** 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 elenagarcia01@vt.edu, then you would save your file as **TransistorModels_elenagarcia01.v**. This is the file you will be editing to implement your circuits.

***WARNING**: Questa has a tendency to save files without extensions, even when you choose "Save As" and select a file type. When you save files in Questa, 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.*

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. Do not edit the testbench file. In this course, if a testbench is required, it will be provided for you. In ECE 3544 you will learn to write your own testbenches.

Create a new project in Questa. (See the lab manual section on Questa.) Add your renamed transistor models file (e.g. TransistorModels_elenagarcia01.v) and the file **tb_TransistorModels.v** to the project.

***WARNING**: When you create folders and projects in Questa, keep in mind that the names of files and folders should **not** include spaces.*

***WARNING**: Normally you can open a Verilog source file in Questa 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 Questa by choosing **File > Open**, and then choosing any file in the active Project. Once one file is opened, others will open in Questa by clicking on their names in the Project window.*

When you first open the file that will contain your transistor models in Questa, remember to complete the included header block so that it contains the appropriate information. See the lab manual section *Verilog File Expectations*. An example is included 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.*

Recall**: CMOS means complementary pairs of n-type (NMOS) and p-type (PMOS) MOSFETs. **A single transistor will be either nmos or pmos, not cmos.