LAB 1 – Drawing Basic Circuits

Goals

Learn to design simple combinatorial circuits and draw their schematics.

To Do

- Design a 4-bit equality comparator circuit that has two 4-bit binary inputs (A and B) and outputs a logic-1 if both inputs are equal.
- Design a 1-bit comparator circuit that has two 1-bit binary inputs (A and B) and outputs a logic-1 on first output port if $A>B$, and outputs logic-1 on second output port if $A=B$, and outputs logic-1 on the third output port if $A<B$.
- Follow the instructions. Paragraphs that have a gray background, like the following one, denote descriptions that require you to do something.
- To get graded for the in-lab assignments (70% of the total lab grade), you need to show your work to one of the assistants. You need to show your work when you are done with all tasks in this manual. The assistants will enter your grade on Moodle, which you can later check.
- There are more questions in the lab report that you will need to submit to get the rest 30% of your total lab grade.

Introduction

Throughout this course, you will learn to design digital circuits. For later exercises, we will program digital circuits in the Verilog programming language using the Vivado programming suite provided by Xilinx. Vivado allows us to design our circuit, convert our design idea from a description into an actual circuit that can be implemented on the FPGA board (synthesis), allocate resources on the actual FPGA, connect these to implement our circuit (placement and routing), and download the final configuration to the FPGA itself (programming).

Most modern designs are described using hardware description languages like Verilog or VHDL (we will use Verilog in the next few weeks). However, in this exercise, we will first learn to design simple circuits by drawing them on paper.

Part 1. The 4-bit Equality Comparator Circuit (2 Pts)

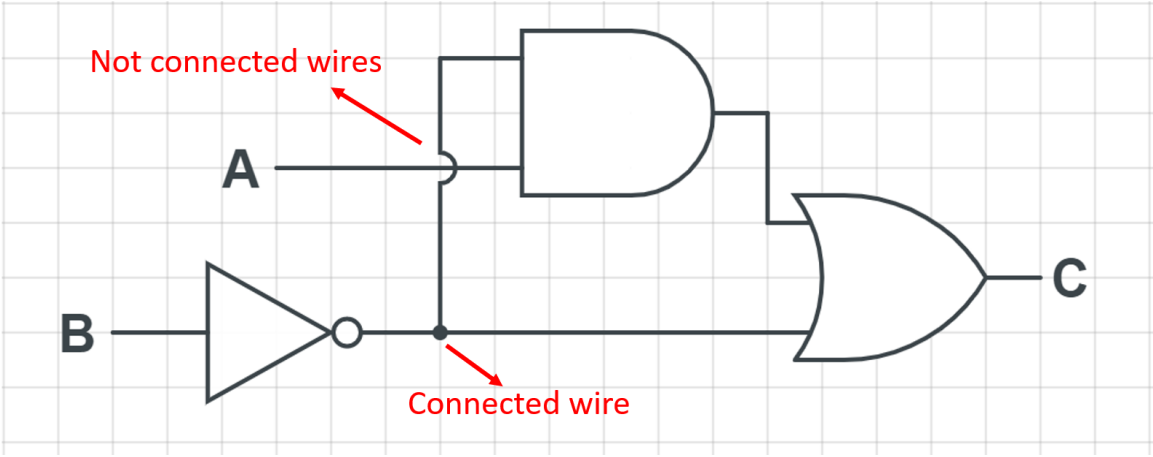
As we have not yet covered how complex circuits can be designed, we start with a fairly simple circuit that we should be able to come up with using a few simple gates. As the name implies, a comparator accepts two values and tells us when the two values are the same. we will use two 4-bit numbers and design the circuit so that the output is logic-1 if both 4-bit numbers have the same value, and logic-0 if they differ.

Sketch the circuit schematic of the comparator using basic 2–input logic gates on paper using a pen. Call the 4-bit inputs A and B, and the output EQ. To make things easier consider a two-step approach: First, compare A and B bit by bit. In the second step,

combine the results of the first step so that EQ is logic-1 only if A and B have the same value.

It is always good practice to have a circuit sketch before beginning to work on the computer. In this case, the circuit is very simple, and the advantage of doing so may not be immediately obvious, but later you will see that having a diagram of the circuit makes life easier.

Below, we provide an example circuit schematic to show how we expect you to label the wires and show the connected/not connected wires.



Part 2. A More General Comparator (2 Pts)

Design a circuit that receives two 1-bit inputs A and B, and:

- sets its first output (O1) to 1 if $A>B$,
- sets the second output (O2) to 1 if $A=B$,
- sets the third output (O3) to 1 if $A<B$.

Sketch the circuit schematic of the 1-bit comparator described above using basic 2–input logic gates on paper using a pen.

Part 3. Designing Circuits Using Only NAND gates (3 Pts)

In the lecture, you learned that it is possible to realize any boolean function with only 2-input NAND or 2-input NOR gates. Re-draw the schematic of Part 2 with only 2-input NAND gates. Use as few gates as possible.

Hint – use DeMorgan's Theorem to express the combinatorial logic in terms of NAND operations only.

Lab Report

You will find more questions on the lab report form. You need to submit the report to be able to get the rest 30% of your overall lab grade.

Last Words

Drawing schematics (either by hand or automatically) was the only way to describe circuits before the advent of hardware description languages. For smaller circuits like the one we have designed in this lab, it is not very difficult to draw schematics. For larger circuits however, the amount of work increases significantly.