DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
UNIVERSITY OF BRITISH COLUMBIA
EECE 353 – Digital Systems Design

Lab 1: Combinational Logic

IMPORTANT:
Make sure you read the labinfo.pdf handout posted on the Connect site first to acquaint yourself with the overall lab format.

In this lab, you will become familiar with the software and hardware we will be using in the labs, and use the hardware and software to implement a simple combinational logic circuit. You will also be re-acquainted with VHDL, which most of you will have been introduced to in EECE 259. Even if you don't remember EECE 259 well, you'll still be able to do this lab if you follow the instructions in this handout.

We will be using two pieces of software for most of this course: Quartus II, which is produced by Altera, and ModelSim, which is produced by Mentor Graphics. There are several versions of ModelSim available; we will be using one that has been modified by Altera for use with Quartus II. You can download these programs as described below (they are free), or use them on the departmental computers in the lab.

PHASE ONE: GETTING READY

TASK 1.1: The first step is to install Quartus II and the Altera version of ModelSim on your home PC or laptop, or find a place you can run the software (it will be available on most of the computers available throughout the department).

Installation instructions can be found in the "Digital System Design using Altera Quartus II and ModelSim" tutorial document located in the Lab 1 folder on Connect.

You should use Version 13.0 SP1 for this course (*do not use Version 13.1 or later*, *since it does not support the device you will be targeting*).

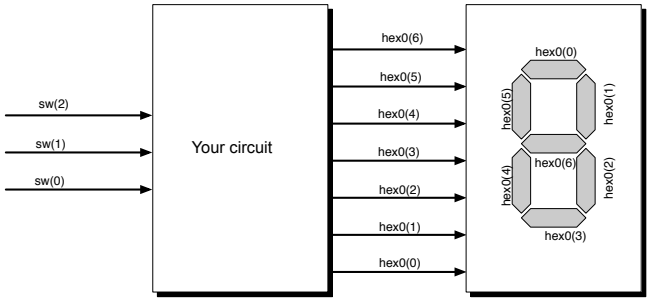
If have an older version of the software from EECE 259, it is a good idea to upgrade to Version 13.0 SP1. Even if you installed 13.0 SP1 last year, Altera is continuing to make "bug fixes" to 13.0 SP1, so you may find it advantageous to re-download and re-install (particularly if you are using Windows 8).

TASK 1.2: Work through the rest of the "Digital System Design using Altera Quartus II and ModelSim" tutorial document located in the Lab 1 folder of Connect. This will introduce you to Quartus II and ModelSim, which are the tools we will use throughout the course. Even if you have used Quartus II, there is a good chance that Modelsim is new to you, so it will save you a lot of time if you work through the tutorial carefully.

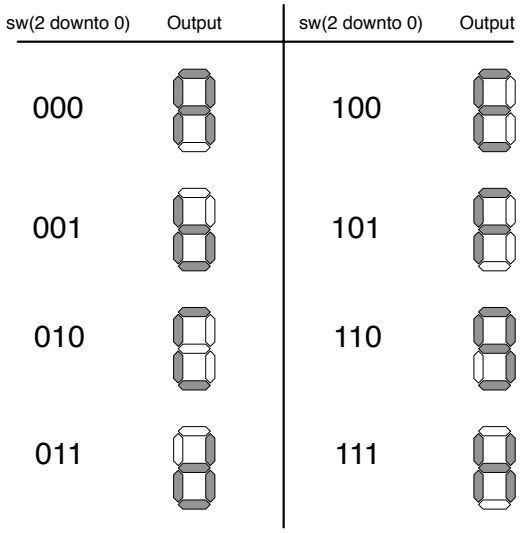
PHASE TWO: SIMULATION

TASK 2.1: First, you are going to design and simulate a simple combinational circuit using VHDL. Don't worry if you have forgotten VHDL from EECE 259; you will be given skeleton files with most of the hard work done. If you want to review material from EECE 259, check out Videos 1, 2, and 3 on the Connect Web site; these are a very brief review of some of the basics that I hope you remember (we will also review this material in class, but the videos will let you get ahead if you want).

The circuit you will design is a combinational circuit with three inputs named *sw(0), sw(1)* and *sw(2),* and seven outputs named *hex0(6)* downto *hex0(0)*. The inputs will be connected to switches, and the output will be connected to a LED hex display as shown below.



The operation of this circuit is that it will convert a three bit binary number into seven signals that control an LED display, as shown below:



Each input to the LED (output of your circuit) controls one segment; if the input is **0**, the segment is **on** while if the input is **1**, the segment is **off** (this is called active-low signaling). The correspondence between LED input signals and the various segments in the display are shown in the diagram on the first page of this handout. So, for example, if hex(0) is a 0, then the top horizontal segment will be on.

To specify this circuit, you should download the skeleton files *converter.vhd* and *converter_tb.vhd* from the course site (in the "Labs/Lab 1" folder). The first file, *converter.vhd* is where the functionality of the block will be described. The second file, *converter_tb.vhd* is a testbench file contains the instructions the simulator will use to exercise (stimulate) your design during simulation. In this lab, the complete *converter_tb.vhd* is given to you; you won't have to modify the testbench file. All of your changes will be in the file *converter.vhd* .

The only part of *converter.vhd* you need to modify is the part that describes the logic functionality of your block. If you look near the end of your file, you will see a block of code that looks like this:

```
case SW is
    when "000" => HEX0 <= "0001000";
    when "001" => HEX0 <= "0000011";
    when "010" => HEX0 <= "insert a string of 7 bits here";
    when "011" => HEX0 <= "insert a string of 7 bits here";
    when "100" => HEX0 <= "insert a string of 7 bits here";
    when "101" => HEX0 <= "insert a string of 7 bits here";
    when "110" => HEX0 <= "insert a string of 7 bits here";
    when others => HEX0 <= "insert a string of 7 bits here";
end case;
```

The first thing to note is that SW is used as a short-form for the collection (or 3-bit *bus*) of signals SW(2), SW(1), and SW(0). When we say SW is equal to "011", we really mean that SW(2) is equal to "0" (the first digit of "011"), SW(1) is equal to "1" (the second digit of "011"), and SW(0) is equal to "1" (the third digit of "011"). It is not a mistake that the indices count "down" through the bus (ie. the first digit is the highest numbered or *most-significant* bit); this is a common design pattern you learned about last year. Similarly, we use HEX0 to refer to the 7-bit bus consisting of signals HEX0(6) downto HEX0(0). So, if HEX0="1000000", then HEX0(6) is equal to 1 (the first digit of "1000000"), HEX0(5) is equal to 0 (the second digit of "1000000" and so on).

The second thing to note is the structure of the CASE statement itself. The first line indicates the selection criteria for this CASE statement (this will become clear in the following discussion). The remaining lines in the case statement indicate an action that should occur for each value of the selection criteria. So for example, the first "when" line indicates that when the selection criteria SW is "000", the output HEX0 should be assigned the value "1000000". If we compare this to the second figure in this handout, we can see that when SW is 000, the output should display a pattern with the lower horizontal segment off, but all other segments on (this looks like a "A" on the display). Remembering that a segment is "off" when its control line is 1, and "on" when its control line is 0, we can see that the pattern "0001000" will turn off the middle segment (bit HEX0(3)) and turn on all the other segments, causing the display to show a "A". Similarly, the second "when" line indicates that the SW is "001", the output HEX0 should be assigned a value "0000011". From the handout figure, this means that bits HEX0(6), HEX0(5), HEX0(4), HEX0(3), and HEX0(2) should be on, while HEX0(1) and HEX0(0) should be off. This will display a "b" on the display.

The remaining 6 lines in the case statement correspond to each of the other 6 possible values of SW. Your task is to fill in the value assigned to HEX0 for each of these 6 lines (to be more explicit, replace each string "insert a string of 7 bits here" with 7 bits corresponding to the display pattern for each line). Remember that each string of 7 bits should be surrounded by quotes, and don't forget the semicolon at the end of each line. Note that the selection criteria for the last line is labeled "others" rather than "111"; we'll talk more about the reasoning behind this later in the course.

TASK 2.2: Using ModelSim, simulate your design (as you learned when you worked through the tutorial document). The supplied testbench *converter_tb.vhd* will apply each of the 8 possible input values to the inputs (one value every 5 ns). Using the waveform viewer, you can determine whether your design is correct. Note that there are two possible errors you might run into here. First, if you have made syntax errors, (such as forgetting the quotes or semicolons, or getting the wrong number of digits, for example) you will get error messages saying your design could not be loaded or simulated. In that case, you won't be able to view a waveform; you have to fix the syntax error before proceeding. Once your syntax is correct, and you can simulate your design, it is possible that you see the wrong pattern of 1's and 0's in your output waveforms. In that case, check the assignment statements to make sure you didn't write in the wrong values of 0's and 1's, or didn't accidently swap the order of bits within a bus.

➔ **BE PREPARED TO DEMO YOUR SIMULATION TO THE TA FOR 4 MARKS.**

PHASE THREE: REAL HARDWARE

In this part of the lab, you will download the circuit you designed earlier onto the Altera Cyclone II FPGA on your DE2 board. The input and output pins of this FPGA are tied to the various lights and switches and other devices.

TASK 3.1: Compile your design using Quartus II and be sure you don't have any synthesis errors. If you do, you will need to fix them. For this lab, it is most likely that any errors would have been uncovered during your ModelSim simulation, however, in future labs, there is a large class of errors that might be encountered here (unsynthesizable code such as "inferred latches"). If you are getting errors at this point, talk to your TA.

TASK 3.2: Read Section 4 of the tutorial document, which describes how to download your design onto the FPGA board. As described in the tutorial download your 7-segment LED Driver circuit you designed

in the preparation. BE SURE that you set the pin assignments before compiling your design (not doing so could damage the board!). If you have any questions about how to do this, please talk to the TA.

TASK 3.3: Multiple Displays. Extend your design so that it displays 4 different digits. To do this, you will have to add extra outputs for displays HEX3, HEX2, and HEX1 as well as extra inputs for a total of 12 switches. **DO NOT SIMPLY COPY YOUR CASE STATEMENT FOUR TIMES!** Instead, use design hierarchy to create and connect four instances of the *converter* component you developed for Task 3.2.

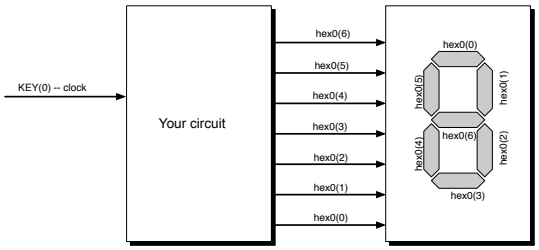
➔ **BE PREPARED TO DEMO THE RESULT OF TASK 3.3 TO THE TA FOR 5 MARKS.**

REMINDER: **TASKS 2.2 and 3.2** MUST BE MARKED TO THE TA BEFORE THE END OF LAB.

PHASE FOUR: CHALLENGE TASK

Challenge tasks are tasks that you should only perform if you have extra time, are keen, and want to show off a little bit. Generally it will be far more work than the other tasks in the lab. This particular challenge task is worth 1 mark.

In this challenge task, you are to create a sequential circuit. Your circuit will now have one input (KEY(0) which is one of the pushbutton switches that we will use as a clock signal) and seven outputs connected to a seven-segment display as shown below. When the circuit first starts, the seven-segment display shows an "A". Each time KEY(0) is pressed, the value on the seven-segment display changes. The first time it is pressed, the display changes to 'b'. The second time, the display changes to 'C', and so on. After the 8th time, the display goes back to 'A' and repeats.



For the challenge mark, you must demo your working circuit on the FPGA board (simulation is not enough). Demo and explain your circuit to the TA.

You will need to review your notes on sequential circuits; Videos 4, 5, and 6 on the Connect site will review the essentials, or pull out your EECE 259 notes.

Hint: Think about feeding the input from your Task 3.2 circuit with the output of a counter.

➔ **DEMO THE RESULT OF THE CHALLENGE TASK TO THE TA FOR 1 MARK.**

REMINDER: TASKS MUST BE DEMOED DURING YOUR FIRST LAB SESSION.